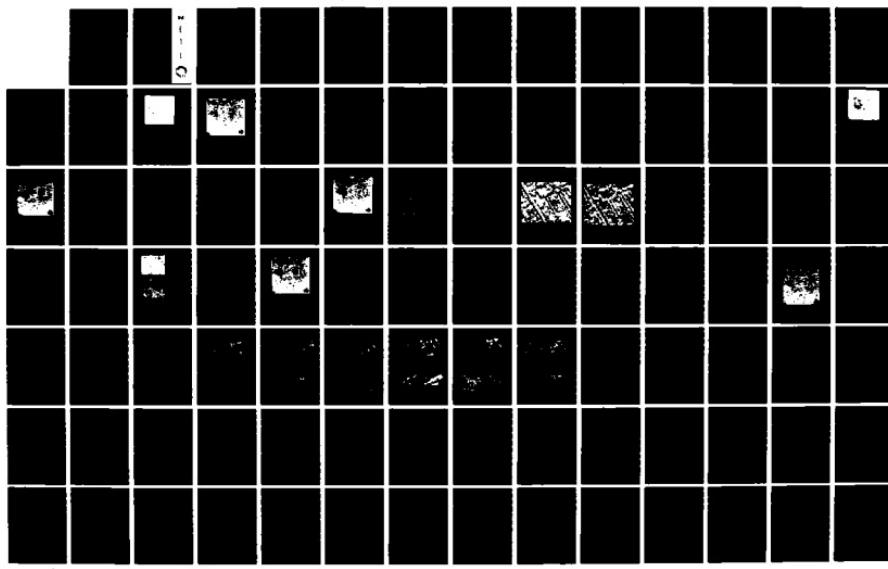


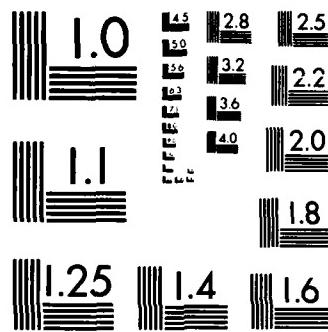
AD-R153 112 FURTHER STUDY OF DIGITAL MATCHING OF DISSIMILAR IMAGES 1/2
(U) L N K CORP SILVER SPRING MD D LAVINE ET AL. FEB 85
ETL-0385 DRCAT6-82-C-0003

UNCLASSIFIED

F/G 12/1

NL



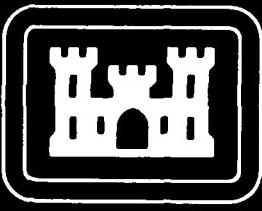


MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A153 112

ETL-0385

2



Further study of digital matching of dissimilar images

David Lavine
Eric Olson
Barbara Lambird
Carlos Berenstein
Daniel Leifker
Laveen Kanal

L.N.K. Corporation
302 Notley Court
Silver Spring, Maryland 20904

February 1985

DTIC FILE COPY

Prepared for

U.S. ARMY CORPS OF ENGINEERS
ENGINEER TOPOGRAPHIC LABORATORIES
FORT BELVOIR, VIRGINIA 22060-5546

APPROVED FOR PUBLIC RELEASE, DISTRIBUTION IS UNLIMITED

DTIC
FILE COPY
2 MAY 2 1985
S



Destroy this report when no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official
Department of the Army position unless so designated by other
authorized documents.

The citation in this report of trade names of commercially available
products does not constitute official endorsement or approval of the
use of such products.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ETL-0385	2. GOVT ACCESSION NO. <i>AD-A153 112</i>	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) FURTHER STUDY OF DIGITAL MATCHING OF DISSIMILAR IMAGES		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report July 1982 - September 1984
7. AUTHOR(s) David Lavine Eric Olson Barbara Lambird		6. PERFORMING ORG. REPORT NUMBER DACA76-82-C-0003
9. PERFORMING ORGANIZATION NAME AND ADDRESS L.N.K. Corporation 302 Notley Court Silver Spring, MD 20904		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Engineer Topographic Laboratories Fort Belvoir, VA 22060-5546		12. REPORT DATE February 1985
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		13. NUMBER OF PAGES 106
		15. SECURITY CLASS. (of this report) Unclassified
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution is unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) edge detection, filtering, match accuracy, point feature matching, .../ registration		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report reviews the L.N.K. registration algorithm and provides an example. The matching of radar and optical imagery was investigated. Special processing of the radar imagery was found to be required and an edge-preserving local filter was successfully applied. A foundation was begun for transformation reliability estimation through the concept of point pattern spaces, as an alternative preferable to least squares. Pattern spaces for translation, rotation, and point reordering were considered.		

PREFACE

This is the Final Report on contract DACA76-82-C-0003 from the U.S. Army Engineer Topographic Laboratories. The work reported here was performed by Dr. Carlos Berenstein, Dr. Laveen Kanal, Ms. Barbara Lambird, Mr. David Lavine, Mr. Daniel Leifker, and Mr. Eric Olson. Mr. David Lavine served as Project Manager. Mr. A.T. Blackburn was the Contracting Officer's Representative.



A-1



Table of Contents

1.0 Introduction	1
2.0 Registration	5
2.1 Basic L.N.K. Matching Technique	5
2.2 Data	9
2.3 Registration Results	14
3.0 Feature Extraction on Imagery	18
3.1 Marr Edge Detector	18
3.2 Connected Sets	25
3.3 Region Growing or Shrinking for Edge Extraction	32
3.4 Local Filtering	44
3.5 Summary	60
4.0 RIPS Conversion	61
5.0 Probabilistic Modelling of Point Patterns	63
5.1 Introduction	63
5.2 Pattern Spaces	64
5.3 Translation	66
5.4 Rotation	69
5.5 Reordering	74
5.6 Registration Reliability Example	75
5.7 Summary	79
6.0 Conclusions and Future Work	82
7.0 References	84
Appendix A - Image Processing Library	A1

1.0 Introduction

This report describes our continuing investigations into the L.N.K. registration procedure. The present study contains both theoretical studies related to algorithm performance, experimentation with optical and radar data and a study of several image processing routines to aid in the extraction of information from radar imagery. The directions of this study were determined, in part, by the directions of previous L.N.K work in this area and in part, by a request by the Engineer Topographic Laboratories (ETL) to explore the feasibility of registering radar data.

L.N.K's theoretical studies were in two basic directions. First, we continued our exploration of error analysis of the registration algorithm which was begun in a previous contract [DACA76-81-C-0004]. In that study, we performed extensive simulations to model the effect of different types of errors on the accuracy of the registration process. In our report on that work, it was pointed out that the noise model used had undesirable characteristics, but at the time, we had no clear idea how to rectify this shortcoming. The registration algorithm extracts points such as road intersections from an image and tries to match these points with a set of corresponding points in a model. We were interested in modelling the effect of perturbing the pattern of points on the accuracy of the algorithm. Considerable literature on this type of modelling in the case of the analysis of correlation accuracy of grey scale images has been developed. The accuracy of the L.N.K registration procedure however breaks down into two parts, the analysis of the accuracy with which interest points can be extracted and the analysis of the effect of the accuracy of interest points on the accuracy of the registration process. In this report, we have focused on the latter

problem.

A basic issue we address in this report is that of defining point patterns, i.e., a pattern consisting of points, in which only the locations of the points relative to one another is considered. We have tried to define a notion of point pattern and define a space in which each point in that space corresponds to exactly one entire point pattern and in which the distance between points in the space corresponds, in some reasonable way to a measure of similarity between different point patterns. Given such a space, one would then like to place a probability measure which could reflect the likelihood of various perturbations of a pattern and then compute the registration accuracy characteristics of the L.N.K algorithm in terms of the probability measure. Since the L.N.K algorithm requires a variety of tolerance parameters, such computations could be useful in tuning the algorithm and determining the reliability of the results. In the present study we develop several examples of pattern spaces which are approximations to the main space we sought. We think the description of the main space can be developed without too much additional effort, but this was not possible with the available funds. We consider this work to be the beginnings of a theory for the analysis of a variety of registration algorithms which is based on feature matching. After developing various models, we illustrate the use of this approach with an application to the problem of determining the reliability of a registration transformation given a measure of goodness of fit of the model and image points under the registration transformation.

Our experimental work on the present contract revolved around the problem of matching a radar image to an optical image. The radar image contained significant perspective distortion, which is not currently taken into account by the registration procedure. In light of this, the performance of the

algorithm may be divided into two questions. First, given perfect extraction of features, is the algorithm sufficiently robust to match the images given the perspective distortion. Second, can good features be reliably extracted.

In answer to the first question, we found that given hand-picked points, the algorithm was still able to register accurately, even with the perspective distortion present in our samples. The extraction of reliable points from the radar image was a much more difficult problem. The characteristics of radar imagery are very different than those of the optical imagery for which our feature extraction algorithms were designed and tuned. In addition, our radar data was taken from an area with a considerable number of man-made features which contributed to the complexity of the resulting image. We found that our existing algorithms for edge and curve detection did not function well in the radar imagery and we sought other means for extracting reliable features.

Two promising approaches to feature detection in radar imagery were investigated. First, we explored the possibility of thresholding the radar image to form a binary image and doing edge detection in a binary image to obtain the required features. This approach was an attempt to deal with the problem of very splotchy looking radar imagery in suburban and urban areas. This approach requires further experimentation but it appears to be promising, especially if used for verification in conjunction with other edge finding methods.

We tried a variety of conventional smoothing techniques, such as mean and median filtering to try to form a more tractable version of the radar image. These approaches were quickly discarded in favor of an edge-preserving filtering technique developed by Lee [1981]. This technique filters by smoothing, but the neighborhood over which the filtering is done is determined

only after a primitive edge detector has been run. The results of this detector are used to determine if there is an edge in a neighborhood of a point whose noise-free value is being estimated. If an edge is detected, then the estimated pixel value for the point is determined only from those pixels on the same side of the edge as the point in question. The results which are displayed in this report indicate a significant level of noise cleaning while preserving the edge structure of the image.

In addition to the work described above, L.N.K., at the request of ETL, aided ETL with the conversion of part of the registration software to run on ETL's Cyber.

In order to effectively study procedures for handling the radar and optical imagery, it was necessary to develop a set of image handling routines providing the capability of various forms of image display, interactive graphics, and a variety of low level image processing routines. A brief description of these utility routines is given in Appendix A.

Section 2 describes the basic L.N.K. registration procedure and gives the results obtained under this study. A more detailed description of the registration algorithms can be found in earlier reports [Stockman et. al. 1981]. Section 3 describes the results of applying various image processing techniques that were tried to clean and extract road intersections from the radar image. Section 4 contains a brief description of the RIPS conversion done at the request of ETL. Section 5 describes the work performed so far in the determination of an intrinsic space for point patterns and an illustration of how this work can be applied to the estimation of registration accuracy.

The work described in this report provides a continuation of our theoretical and experimental studies into the performance of the L.N.K registration algorithm. We wish to thank personnel at ETL for providing us with data for this study.

2.0 Registration

This section contains a brief description of the L.N.K. registration procedure and the results of matching using hand-picked intersection points from both the optical and radar images.

2.1 Basic L.N.K. Matching Technique

This section presents an overview of the basic L.N.K. matching procedure. A more detailed description of the procedure is available in [Stockman et. al. 1981]. The goal of matching is to use automatically extracted features from an image and a model (or two images) to find the best global correspondence between the image and the model.

The L.N.K. procedure is able to achieve this even when many local mismatches of features occur. The three basic steps of the method are

- 1) Primitive features, such as line segments, edge segments, intersections, or high curvature points are automatically extracted. These features may be parameterized, such as by position, by length, or by orientation.
- 2) Assume all features of one type can correspond to one another. For example, an intersection of three lines in one image can correspond to any three-line intersection in the second image. However, only one of these correspondences is the true one. Now, for each of the possible correspondences, find the translation and rotation that maps one feature to the other. Let that translation and rotation transformation be represented by the triple $(\Delta\theta, \Delta x, \Delta y)$. Place a unit of weight in the bin in the three dimensional histogram that represents $(\Delta\theta, \Delta x, \Delta y)$. This process is repeated until all possible feature correspondences have been transformed.
- 3) Locate any prominent clusters of $(\Delta\theta, \Delta x, \Delta y)$ in the histogram. Each cluster represents a set of features in one image that could be mapped to corresponding features in the other image by one particular $(\Delta\theta, \Delta x, \Delta y)$. The best global trans-

formation is defined by the $(\Delta\theta, \Delta x, \Delta y)$ of the largest number of local correspondences.

This method works because the correct transformation will show up as a large cluster, while the wrong transformations will tend to be distributed randomly throughout the histogram. The L.N.K. matching technique can be generalized to handle scale and perspective by adding more parameters to the transformation space [Lambird et. al. 1980].

As an example of the basic technique, suppose an image can be represented by the four directed edge segments shown in Figure 2-1a. The direction of the arrow is determined by making the region on the right of the edge segment the darker one. Suppose the model contains the edge segments in Figure 2-1b. There are 16 possible ways that the edge segments in (a) can be paired with the edge segments in (b). Four of the 16 pairings yield a consistent interpretation -- rotate by $\theta=45$ and translate by $(4.5, 2.0)$. The triple $(\theta=45, \Delta x=4.5, \Delta y=2.0)$ forms a cluster in the three-dimensional histogram while the triples from the incorrect pairings are sparsely distributed. Figure 2-2 shows the 16 possible transformations in the transformation space. The correct transformations are indicated by an '*'. In actual cases, there will be many more than 4 primitive features and not all pairings will be possible (i.e. due to size, shape, or type differences) so the cluster of correct transformations should be even more prominent.

As stated in step 1 of the procedure, many features could be used to perform the transformation. Although we previously have used mostly linear or point features, other types of features may be used.

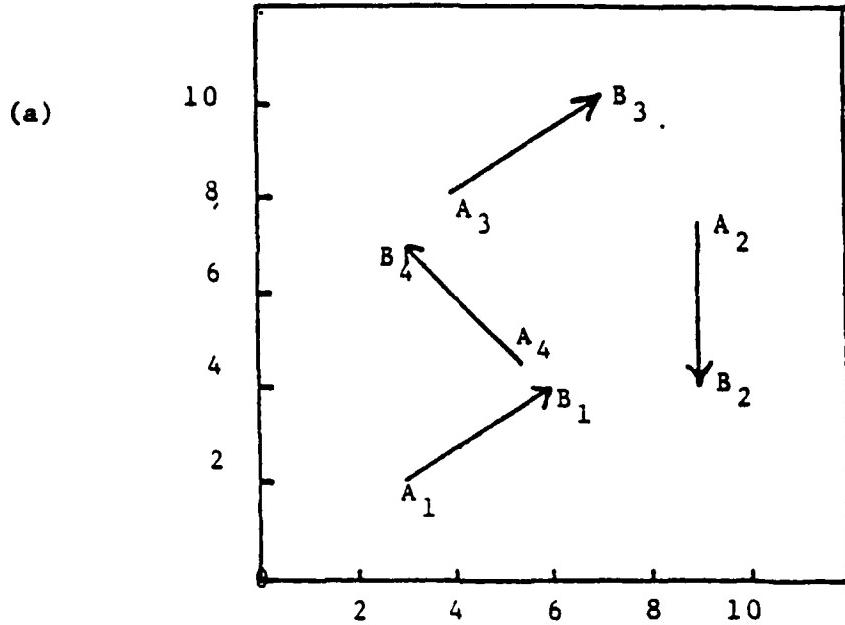
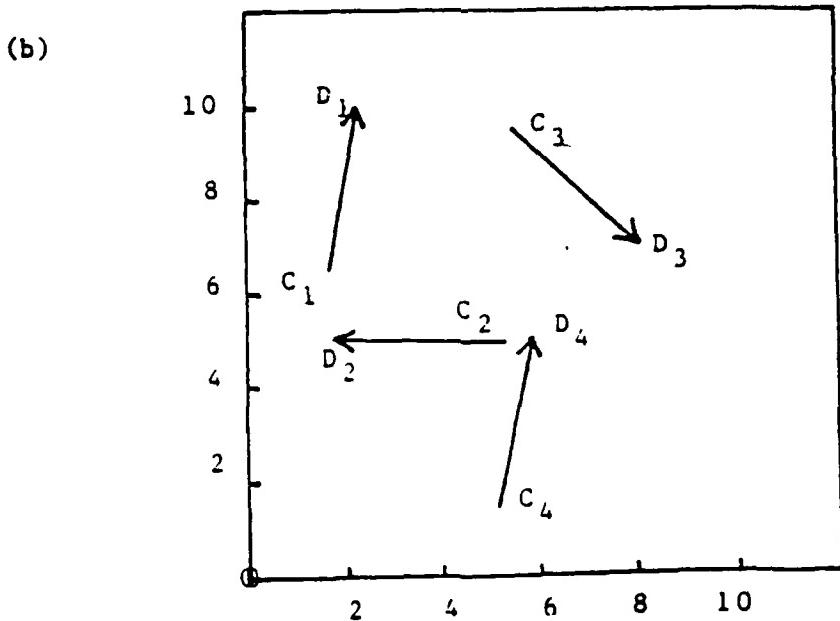


Figure 2-1. Example of basic L.N.K. matching technique. Image elements in (a) need to be rotated 45° and then translated $(4.5, -2.0)$ to be transformed into corresponding map edge elements in (b).



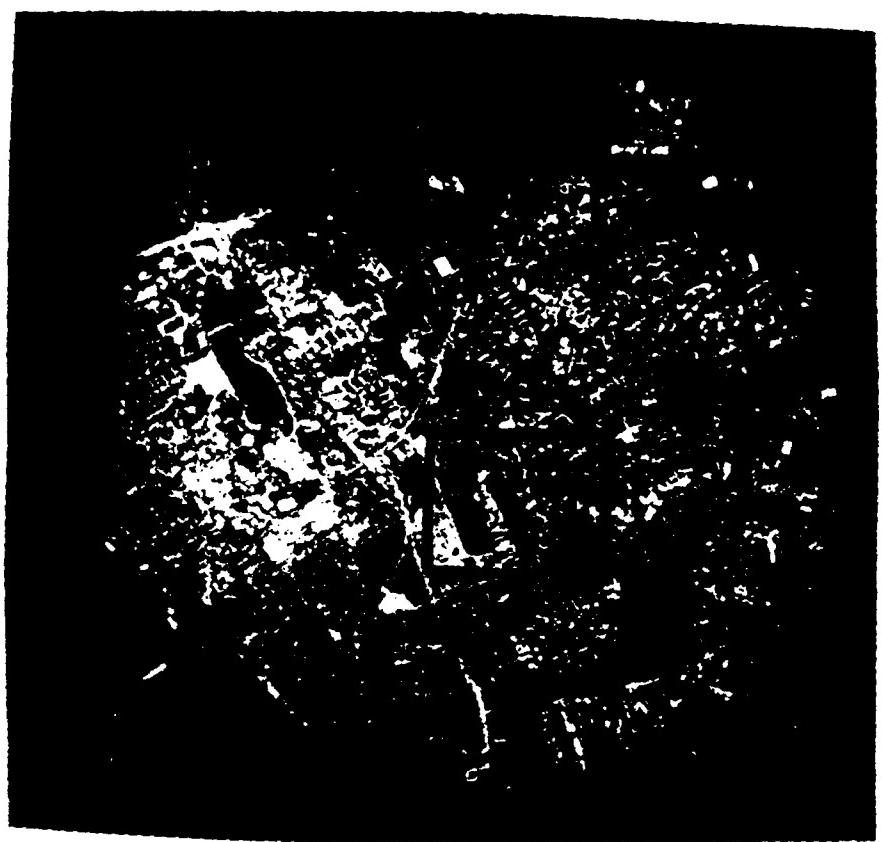


Figure 3-2a. Optical window used for Marr edge operator.

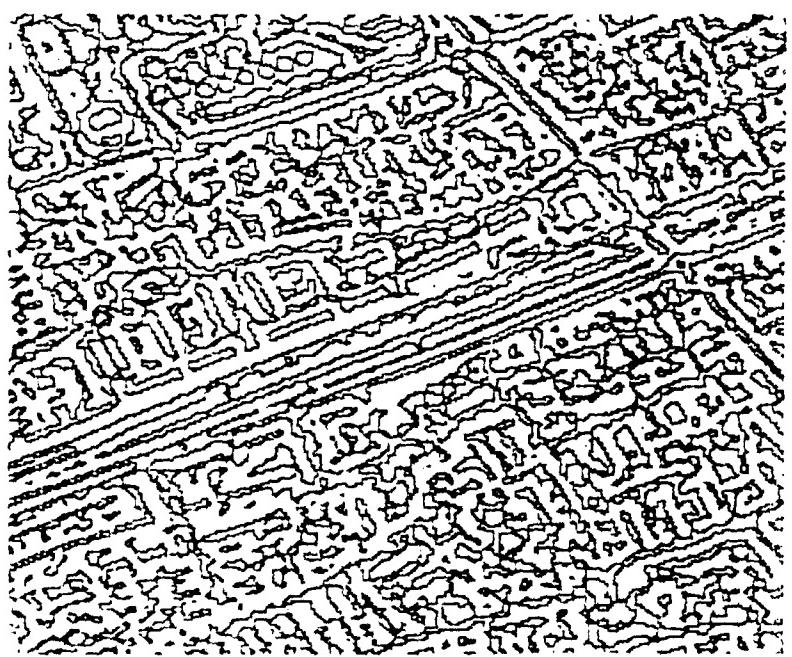
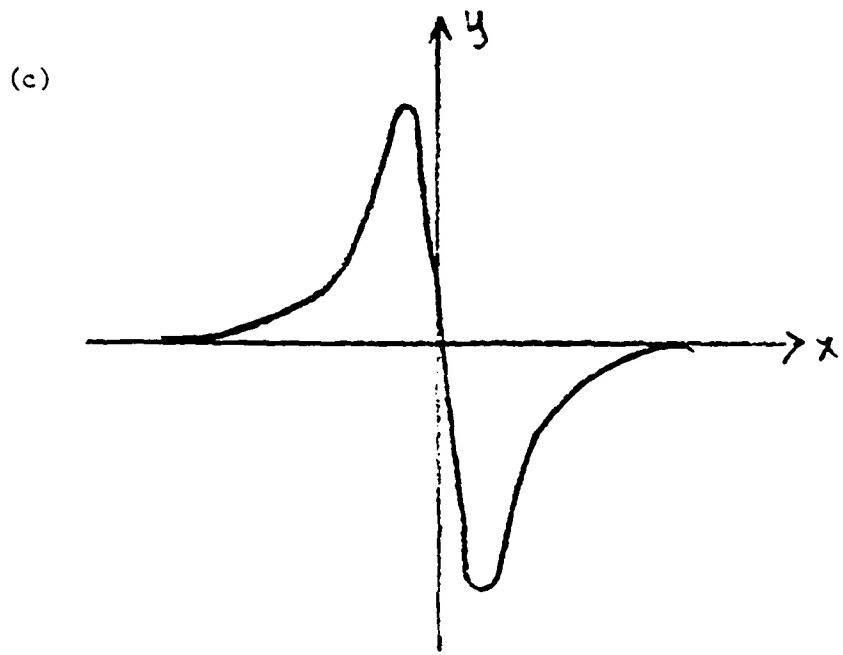
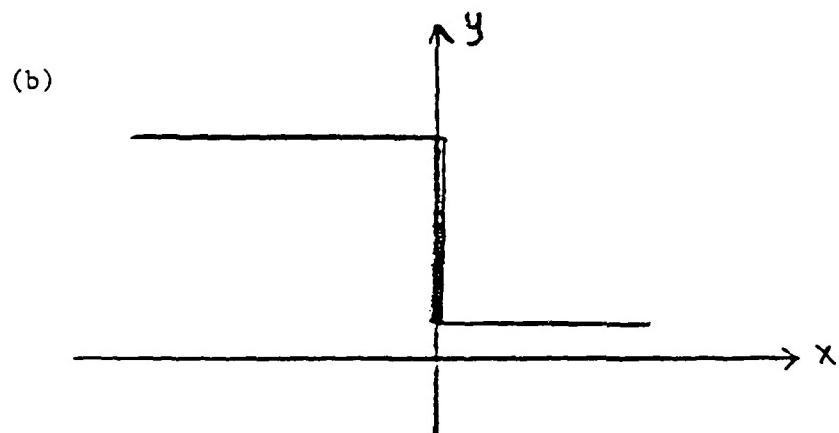
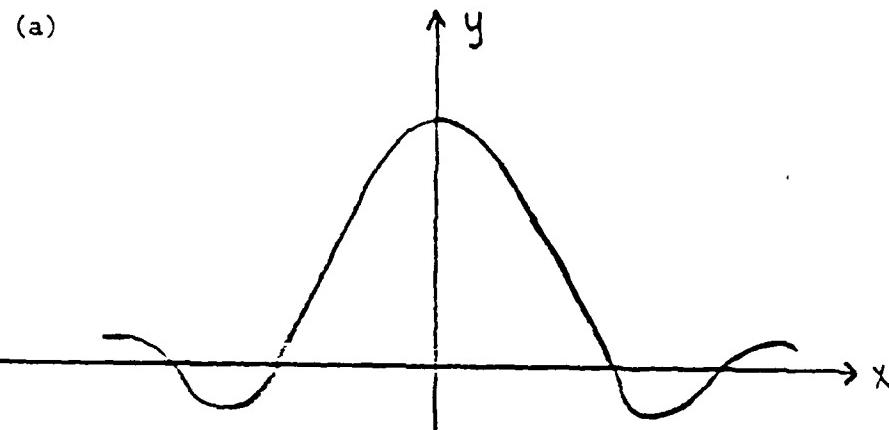


Figure 3-2b. Marr detection on 128 X 128 optical window.
 $\sigma=10$, Mask size = 11 X 11

Figure 3-1. The Marr edge detector (a) convolved with an edge (b) produced (c), where the zero crossing indicates an edge.



Marr [1979] used the Laplacian of a two-dimensional Gaussian distribution convolved with an image as an edge detector. The Laplacian is an approximation to the second derivative of the image. It is parameterized by a variable, σ , so that the Laplacian can be made smooth and band-limited in the frequency domain, and smooth and localized in the spatial domain. By varying σ , objects within a determined size range can be detected and pinpointed in the image.

The two dimensional Gaussian is defined as

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp \frac{-(x^2+y^2)}{2\sigma^2}$$

and its Laplacian is

$$L(x,y) = G''(x,y) = \frac{1}{2\pi\sigma^2} (x^2 + y^2 - \sigma^2) \exp \frac{-(x^2+y^2)}{2\sigma^2}.$$

A cross-section of $L(x,y)$ is shown in Figure 3-1(a). The two-dimensional $L(x,y)$ is found by rotating the graphed curve about the y-axis. The central portion of $L(x,y)$ is negative and is 2 wide. The outer portion of $L(x,y)$ is positive. The peaks of $L(x,y)$ occur at $x = \pm\sqrt{3}\sigma$ from the origin.

A cross-section of an edge is shown in Figure 3-1(b). When $L(x,y)$ is convolved with this edge, the curve in Figure 3-1(c) results. Note that where this curve crosses the x-axis is where the edge occurred in Figure 3-1(b). Therefore edges are found by locating the zero-crossings of the Laplacian convolved with the original image.

One interesting property of the Marr edge detector is it tends to find continuous (gap-free) edge segments. Most other edge detection methods do not do this, and require additional processing to weed out noise points and fill in missing pieces.

The Marr detector was applied to a 256x256 window in the radar image and a 256x256 window in the optical image. The results of the detector along with the windows from the original images are shown in Figures 3-2 and 3-3. A σ

3.0 Feature Extraction on Imagery

After the feature extraction routines that were used in the past to extract intersection points from optical and IR imagery failed on the radar image, we tried other techniques to clean the radar image and extract good edges. This section describes the results of this effort as well as some algorithm experimentation on optical imagery.

Section 3.1 discusses the result of applying the Marr detector to the radar image.

3.1 Marr Edge Detector

The difficulties encountered in trying to extract reliable edges and intersections in the radar imagery using the Hough transform led us to consider other types of edge detectors. Previous L.N.K. work with the Marr [1979] detector led us to consider its use in the present registration application, especially on the radar data. The Marr detector has a parameter that may be changed to control the size of the detected edges. It was thought that this might be useful in detecting the edges on the radar image, since the edge structure is more apparent on a large scale rather than on a small scale.

At the time we were trying the Marr detector, we were considering a variety of other approaches and, as the Marr detector did not produce good results on the radar data, it was abandoned after very limited experimentation. However, this approach might still be quite reasonable on enhanced versions of the radar data such as those obtained through the application of the edge-preserving filtering algorithm [see Section 3.4]. The results of the Marr detector on the optical image yielded a number of identifiable road and man-made structures.

The correct transformation was estimated (see Section 2.2) to be theta=93 , xshift=20, yshift=38. To estimate the quality of the registration performance, we first set an error tolerance of +10 for theta, +20 pixels for each of x and y. Since the image was 512x512, this represented a maximum allowed error of about 4% in each of the x and y coordinates and an error of 3% in theta. Of the 24 runs, 22 were within tolerance. Of the two which were not, one was just completely off and the other was well within tolerance for theta and x-shift and was an additional 1% (5 pixels) outside of tolerance for the y-shift. The average magnitude of the error in each of the estimated parameters was computed for each of the 22 runs within tolerance. The computed mean errors were theta_err_mean=1.05 , x_err_mean=7.55 pixels and y_err_mean=5.59 pixels. The variances for these values are 1.38, 15.6, and 19.6 respectively.

The registration procedure was able to match the radar and optical images to within the selected tolerance in 92% of the runs. Thus the perspective and scale distortions did not cause sufficient differences in the relative locations of the handpicked points in the two images to radically affect the number of matches. The variation of the relative number of good vectors versus the number of bad vectors had no significant effect on the number of registrations which were within the tolerance, since there were only two cases outside of tolerance.

The primary purpose of the above experimentation was to determine the effect of perspective and scale differences between the two images. The effect of these factors on the performance of registration procedure was not significant given that high quality (human) feature extraction was performed.

MAXIMUM MATCH WEIGHT

GOOD	BAD	THETA	XSHIFT	YSHIFT	MTCHWT	MCHROW	NMCHCOL
6	0	93	26	38	4383	4	4
6	3	95	18	36	3738	5	5
6	6	93	26	38	3175	6	6
6	9	95	16	34	2486	6	6
6	12	63	113	64	1127	6	4
6	15	93	8	36	1890	8	5
9	0	95	6	26	5622	4	4
9	3	93	38	63	2416	6	6
9	6	95	26	50	2380	6	6
9	9	93	10	36	4800	11	11
9	12	89	25	55	1128	5	5
9	15	92	14	30	2229	10	10
12	0	95	20	34	4983	10	10
12	3	93	10	36	4393	9	9
12	6	93	8	32	2316	6	7
12	9	92	32	44	3781	10	11
12	12	92	16	34	3433	11	11
12	15	93	10	36	4555	17	17
15	0	93	6	28	4620	10	10
15	3	93	10	34	4244	12	12
15	6	95	14	30	3509	10	11
15	9	95	12	34	4270	14	15
15	12	93	14	32	2922	12	12
15	15	94	16	30	2146	9	9

TABLE 2-1. Registration results on hand picked radar and optical points. Good and bad refer to correct and incorrect abstract vectors. Theta, Xshift, and Yshift are the estimated transformation parameters. MTCHWT is a merit measure for the goodness of vector. Its correct transformation is

$$\begin{aligned} \text{Theta} &= 93 \\ \text{Xshift} &= 20 \\ \text{Yshift} &= 38 \end{aligned}$$

prevented a registration transformation from being determined. In addition, we selected subsets of control points and added spurious points to see if the transformation estimation was effected.

Twenty-four runs of the registration program were made in estimating the robustness of the procedure. In each run, a specified number of vectors connecting pairs of corresponding control points in the two images were used. There were six trials run for each number of good vectors. The number of good vectors in the various trials was 6,9,12,15. For each trial, a set of bad vectors was added to the list of abstract vectors. The bad vectors were obtained by randomly generating pairs of points in the window from a uniform distribution and taking each pair to represent a new vector. The numbers of bad vectors (vectors not resulting from control points) in the various runs were 0,3,6,9,12,15. Thus there was four possible values for the number of good vectors and six possible values for the number of bad vectors resulting in a total of twenty four runs.

The results of the runs are displayed in Table 2-1. The number of good and bad vectors is displayed for each run along with the calculated transformation and the various measures of matching quality [Stockman et. al. 1981]. The matchweight (MTCHWT) is a measure of the total goodness of fit based on the amount vectors which matched have to be distorted to match exactly. The measures NCHROW and NMCHCL refer to the number of image vectors which approximately match model vectors and vice-versa. The registration procedure produced nine clusters of transformations with three transformations given in each cluster. Out of the 27 transformations, we selected the one with the maximum value of MTCHWT as the transformation representing that run. Thus for any pair (# good vectors, # bad vectors), one transformation was selected. The transformations for each of the 24 runs are shown in Table 2-1.

2.3 Registration Results

The registration software was run on the radar and the optical 512x512 windows. After considerable experimentation with the parameters of the various feature extraction routines in the program, we were unable to extract reliable intersections from the radar data. (The routines which are described further in [Stockman et. al. 1981] were the Hough edge detector together with merging and pairing of edge segments and a curve edge detector and tracker.) This led us to explore the other approaches to the processing of radar data described in this report in Section 3. As these approaches still require further work, before they can be used effectively for radar feature extraction, we did not apply the registration procedure to the results. We were unable to obtain more than a few (one-two points in a typical run) points in the radar image which corresponded to intersections in the image and generally none corresponded to intersections detected by the optical program. In light of this, we restricted our registration experimentation to handpicked points in the images to determine the effects of differences in resolution and to perspective distortion.

A set of points was selected by hand from the optical image and a corresponding set was selected from the radar image. In each case, visual inspection was used to determine that the points were taken from the same intersections in each picture. A total of 18 matching points in each image were chosen. The registration program was run in a mode in which a rotation and translation were estimated, but scaling differences and perspective distortion were ignored. The effects of these additional distortions were evident when control points in the two images were superimposed. We ran a series of experiments to see to what extent these additional distortions

the same graphics screen as the untransformed points from the other. We attempted a number of transformations to obtain the transformations which appeared to cause the best overlay. A printout of the points for the best transformation is shown in Figure 2-5. The control points were selected by hand, primarily from intersections.

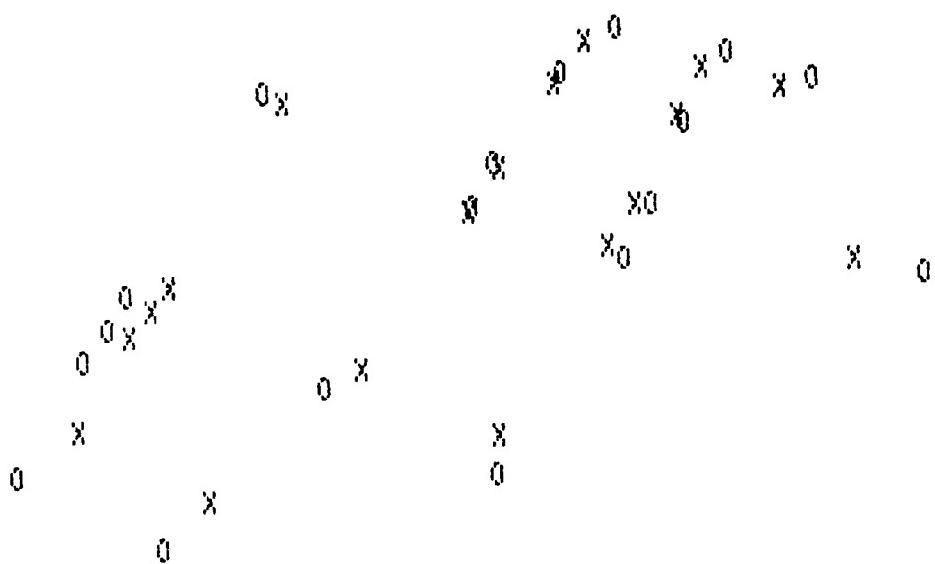


Figure 2-5. Radar and optical handpicked points after best human registration of 512 X 512 images.

X = radar
0 = optical



FIG. 1-4. 512 X 512 radar window used for experimentation.
Upper left hand corner of marked square is row 1,
column 1.



Figure 2-3. 512 X 512 optical window used for experimentation.
Upper Left hand corner of marked square is row 1,
column 1.

2.2 Data

ETL requested L.N.K. to work with the matching of radar to optical data rather than restricting our attention to the theoretical and simulation based studies stated in the contract. Unfortunately, no data was available during the early stages of the contract. Under suggestion from ETL, we obtained a SEASAT image covering the Washington, D.C. area for the registration study. ETL provided an optical image of part of the Washington area, but the difference in scale between the optical and the radar image was large and the perspective distortion in the optical image forced us to search for another image. L.N.K had an optical image of the D.C. area on a transparency and we attempted to digitize it in our laboratory. The quality of the transparency together with limitations in the quality of our digitizing apparatus led us to consider other sources of data. Soon after this problem arose, we were informed by ETL that both L-band radar and optical data of the same area were available.

The radar and optical data were of an area in Germany. The resolution of both sensors was approximately 3 meters, but the difference was discernible in comparing distances in the images. We selected a 512x512 window from each of the radar and optical images. Perspective distortion was quite evident in the full radar image from which our window was selected, though no quantitative description of this distortion was available. The optical and radar images are shown in Figures 2-3 and 2-4. Ground truth for the rotation and translation relating the images was not available from ETL, so we developed a program to help us estimate the transformation. The program allowed the user to specify a set of control points taken from each of the images, and specify a transformation and then displayed the transformed points from one image on

Edge Elements		Endpoints				Transformation		
i	j	A	B	C	D	θ	xs	ys
1	1	3.0,2.0	6.0,4.0	1.7,6.4	2.3,10.0	0.82	1.1	2.8
1	2	3.0,2.0	6.0,4.0	5.3,5.0	1.8,5.0	2.55	8.9	5.0
1	3	3.0,2.0	6.0,4.0	5.5,9.5	8.0,7.0	4.91	3.0	12.1
1	4	3.0,2.0	6.0,4.0	5.1,1.5	5.8,5.0	0.79	4.4	-2.0
2	1	9.0,7.5	9.0,4.0	1.7,6.4	2.3,10.0	2.98	11.8	12.3
2	2	9.0,7.5	9.0,4.0	5.3,5.0	1.8,5.0	4.71	-2.2	14.0
2	3	9.0,7.5	9.0,4.0	5.5,9.5	8.0,7.0	0.79	4.4	-2.2
2	4	9.0,7.5	9.0,4.0	5.1,1.5	5.8,5.0	2.94	15.4	7.1
3	1	4.0,3.0	7.0,10.0	1.7,6.4	2.3,10.0	0.82	4.8	-2.0
3	2	4.0,3.0	7.0,10.0	5.3,5.0	1.8,5.0	2.55	13.1	9.6
3	3	4.0,3.0	7.0,10.0	5.5,9.5	8.0,7.0	4.91	-3.1	11.9
3	4	4.0,3.0	7.0,10.0	5.1,1.5	5.8,5.0	0.79	7.9	-7.0
4	1	5.5,4.5	3.0,7.0	1.7,6.4	2.3,10.0	5.33	-5.2	8.3
4	2	5.5,4.5	3.0,7.0	5.3,5.0	1.8,5.0	0.79	4.6	-2.1
4	3	5.5,4.5	3.0,7.0	5.5,9.5	8.0,7.0	3.14	11.0	14.0
4	4	5.5,4.5	3.0,7.0	5.1,1.5	5.8,5.0	5.30	-1.7	3.6

Figure 2-2. Transformations of all 16 possible combinations of image and map vectors from Figure 4-1. A cluster (indicated by '*'s) was formed at ($\theta=0.79^\circ$, $\Delta x=4.5$ pixels, $\Delta y=-2.0$ pixels).



FIGURE 4. A similar window used for Marr edge operator.

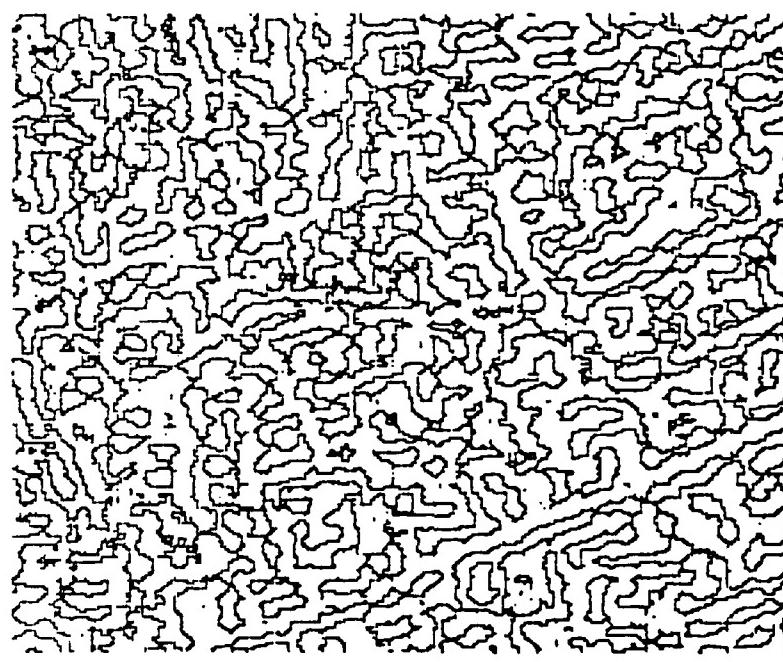


Figure 3-3b. Marr detection on 128 X 128 radar window.
 $\sigma=10$, Mask size = 11 X 11

value 10 was used with an 11x11 Marr mask. Several of the major roads in the optical image show up quite clearly in the Marr results and many small regions in the edge output correspond to regions detectable by eye in the optical picture. The radar edge detection output appears to contain very little useful information apart from one relatively long string of relatively straight edges which are aligned. Given the output of the Marr detector on the radar image, we decided to forgo further work with the Marr detector until we were able to find some means of preprocessing the radar data to remove much of the noise.

3.2 Connected Sets

The quality of the radar image led to extremely poor results in our initial attempts to extract edges using the edge extraction procedures in the L.N.K. matching procedure. These problems occurred with both the Seasat imagery and the radar imagery provided by ETL. In order to deal with this problem, we performed some experiments in low level image processing to determine if alternate means for edge extraction were more suitable for the radar image.

We noted that roads in the radar images often consisted of sequences of blobs of varying sizes scattered along the road sites. In our initial work with the Seasat image of the Washington, D.C. area, the pixels were approximately 25 meter resolution, so only the largest roads, such as the Beltway, appeared and even these were at most a few pixels wide and were often highly distorted in shape. After spending a considerable amount of time examining the pixel values and comparing against the corresponding image locations, it appeared that a large amount of information might be available in a binary image, if proper thresholding was done. Since the appearance of edges in the radar images were quite different than that in an optical image, we felt it would be worth while to spend time trying some elementary operations on the radar image to determine their usefulness.

In the Seasat image of Washington, D.C., many of the roads appeared to have the bulk of their grey levels in the range of 80-100 on a scale from 0 to 255. This was ascertained through a set of thresholding experiments.

An interactive thresholding program was developed which displayed a grey-level histogram on our graphics terminal and permitted all pixels below one threshold to be set to one value and all pixels above another threshold to

be set to a possible different value. Pixels with intermediate values were unchanged. A routine for reading specified windows from a disk file and rescaling the pixel values were required since our display monitor only permitted 64 grey levels while the imagery contained 256 grey levels. The grey level rescaling program allowed all pixel values below a specified value to be set to zero and all pixel values above a specified value to be set to 63 and all other values were linearly interpolated. Simple rescaling by dividing all grey levels by four was not always satisfactory, since much of the significant variation in grey levels was confined to a small portion of the 0 to 255 range in some of the windows.

Once we decided upon thresholds which appeared to delimit road pixels, values were selected and a binary image was formed. The road pixels tended to have values in the range from 4 to 24 out of a range of 0 to 63. The window in the radar image is shown in Figure 3-4. All pixels in the range 4-24 are set to black and all other pixels to white. The long, relatively straight white areas tend to be buildings next to the roads. Thus thresholding on the road pixels tends to bring out the roads by bringing out the buildings near the roads.

As much of the major road structure of the radar images still appeared clear in the binary images and the grey-level routines for edge detection were not doing well, we next tried to clean up the binary image. We noticed that many small isolated regions appeared to clutter up the image. We next ran the connected components algorithm [Rosenfeld and Kak 1976] to see if it could be used to clean up much of the noise. An algorithm was implemented for computing connected components in the binary image. Connected components of either the foreground or background could be computed and any region containing less than a specified minimum number of regions could be converted



Figure 4-1. 125 x 125 pixel window used for annealed component
drawing (photograph).

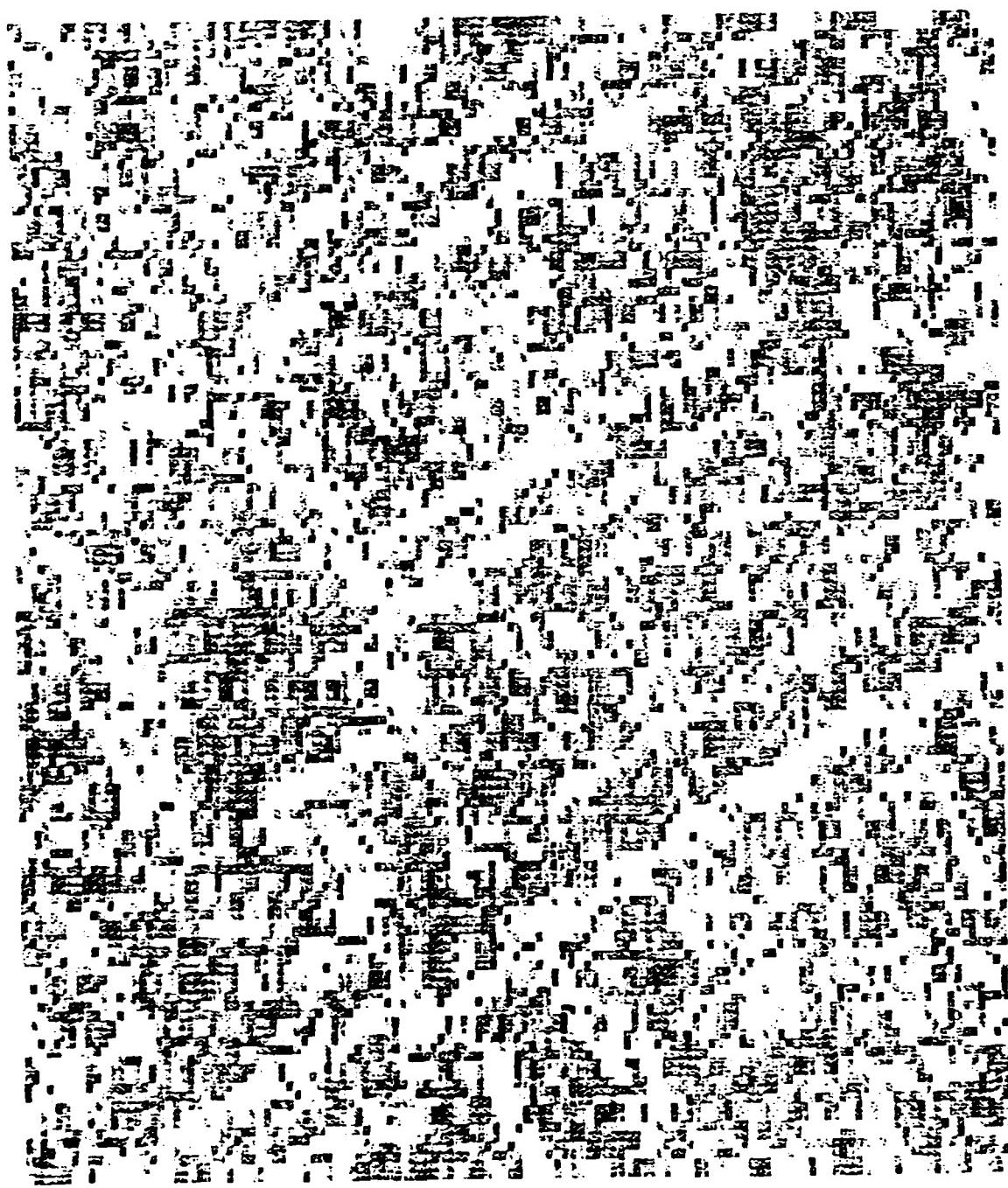


Figure 3-4b. 128 X 128 radar window used for connected component growing (graphics printout).

into the opposite type of region. A set was considered to be connected if it was 4-connected. Thus two pixels are regarded as being adjacent if one is immediately above, below, to the left or to the right of the other.

In Figure 3-5, we see the results of running the connected components algorithm on the binary image. Any 4-connected white region was converted to black. While several relatively long straight areas can be discerned in the image, which do correspond to roads, considerable smoothing is still required. The algorithm was run a second time, on the results of the first run. On this run any black region in Figure 3-5 containing fewer than 50 pixels was converted to white. The results of this are shown in Figure 3-6 where black and white are reversed from Figure 3-5, i.e. the road is now white. This did not appear to provide a significant improvement over the previous image. Connected components do not appear effective in delineating the roads, though they might be effective in conjunction with further processing.

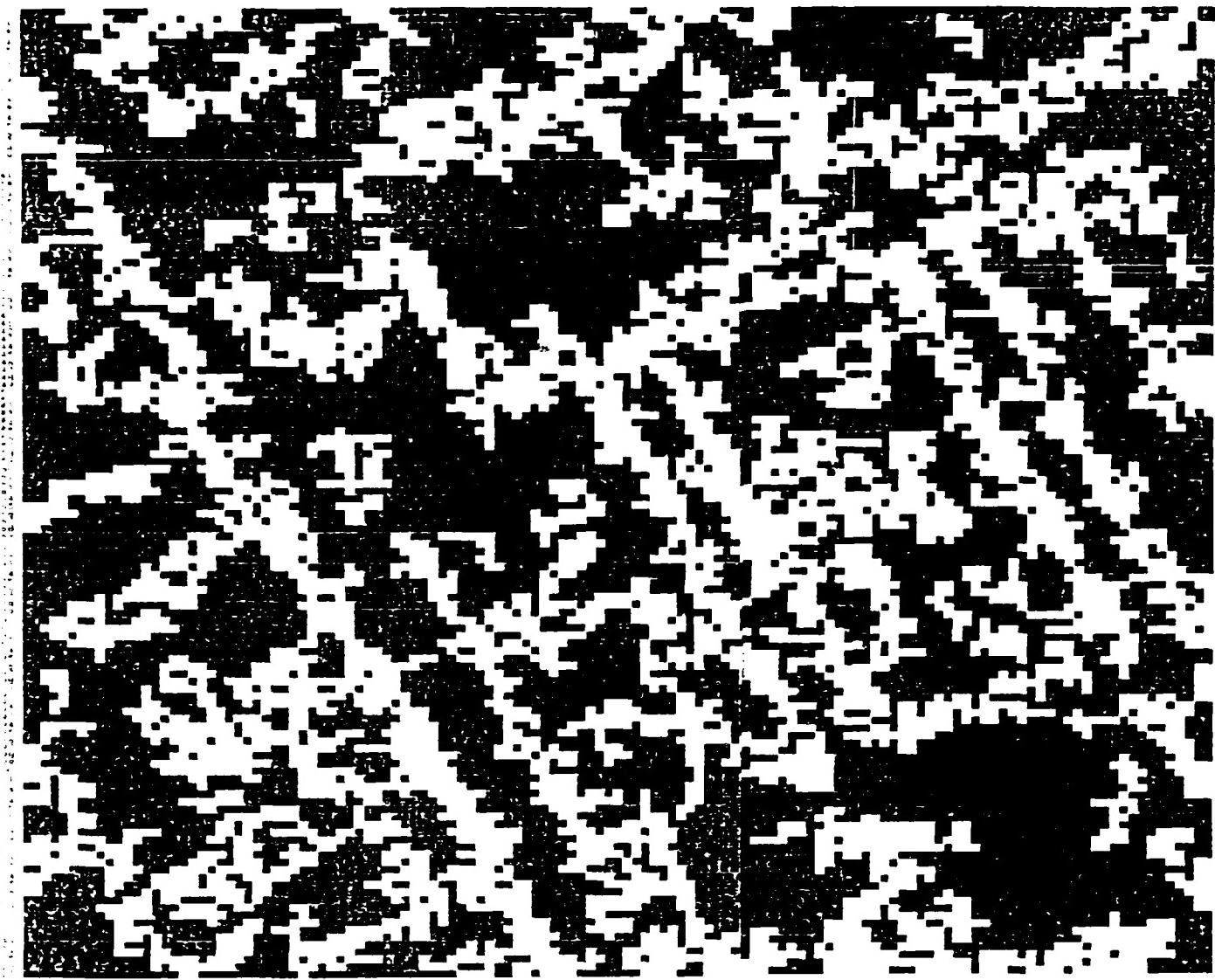


Fig. 4. Result of connected components shrinking in regions with gray levels not in the range 4 - 24. Regions with less than 10 pixels are converted to opposite binary gray level. The background is black.



Figure 3-6. Result of connected components, shrinking black regions in Figure 3-5. Regions with less than 50 pixels are changed to white. Finally black and white are reversed so road is white.

3.3 Region Growing or Shrinking for Edge Extraction

The problems in extracting reliable edges in the radar imagery, described in Sections 3.1 and 3.2, led to a study of the information content in binary images. While the connected components work described in Section 3.2 appeared to clean the images somewhat, we thought other possibilities should be explored. One characteristic of the radar images was the presence of many highly irregularly shaped blotchy areas which appeared to cause considerable trouble in the gradient directional computations for the Hough transform and the curve tracking routines [see Section 2]. One approach to the image cleaning problem which has generated considerable interest in the last several years is the use of boundary shrinking [Serra 1982]. For simplicity of interpretation, we considered the binary case in the present study.

Thresholding was performed on the ETL radar image as described in Section 3.2 to obtain a binary image. After experimenting with a variety of thresholds, it was observed that grey levels in the range of 15 to 60, out of a range from 0 to 255, contained a great deal of information about the road structure in the image. The image was then thresholded to yield a binary image in which pixels in the range from 15 to 60 were set to a value of 1 and all other pixels were set to a value of 0.

The irregularities in the boundary shape of many of the roads led us to consider boundary smoothing operations. Consider the case of a binary image. One simple smoothing operation is to remove all boundary pixels from the object (assuming the object is taken to be 8-connected). In portions of the object where the boundary is very irregular, a much larger percentage of the pixels in the object tend to be boundary pixels. Thus this approach removes more of the object in the vicinity of irregular boundaries. As an alternate

approach, a pixel with value one and at least one 4-neighbor with value zero can be set to zero. This produces a less severe reduction of the object, while still following the general strategy of reducing the object size in areas where the boundary is extremely irregular. This same approach can be used to reduce the size of the background. In addition it can be applied iteratively to either object shrinking, background shrinking, or some combination of the two. The general study of such sequences of operations is referred to as mathematical morphology [Serra 1982]. In that area, a broader range of masks, in addition to four and eight neighbors of a pixel, is considered.

After obtaining a binary image and smoothing it, we were next interested in means of extracting edges from a binary image, especially for the radar image, since more conventional methods appeared adequate for optical imagery. Many conventional edge detection processes may be thought of as consisting of two major steps. First local evidence of edge presence and direction is obtained using some type of operator such as a gradient. Second, the local evidence is combined into global information about the presence of edges. In some cases, it would be desirable to directly use global information to locate edges, but the cost is often prohibitive.

One natural global approach is the use of templates for edge fitting. The Hough transform is a widely used variation on template matching in which the matching is done in a transformed domain rather than the original domain for efficiency. Since the binary images appeared to contain significant edge information, we tried to see if something like an edge template could be used. Rather than using a simple convolution mask, we attempted to make better use of the characteristics of the data.

Our basic approach to binary edge finding was to place a window over a

portion of the image and measure the edge content within the window. The measure of edge content consisted of two components, one which reflects the difference in average brightness on two sides of the suspected edge position and a second which measures the distribution of ones and zeros on each side of the edge. The first measure was considered the primary basis for considering the presence of an edge and the second was to help decide if the area was sufficiently noisy to make the presence of an edge dubious. The window used for edge detection was a parallelogram which was divided into a set of parallel digital lines which filled the window. The lines were drawn using the Bresenham line-drawing algorithm [Newman and Sproull 1973] used in computer graphics.

The purpose of the algorithm was to determine evidence of an edge transition between each pair of adjacent lines in the window. For each pair of adjacent lines in a window, the magnitude of the number of pixels with value one in one line minus the number of pixels with value one in the other line was computed and divided by the length of the line. For a perfect edge in which all pixels to one side of the edge had value zero and all pixels to the other side had value one, this measure gives a value of one. In a homogeneous region, the measure gives the value zero and otherwise it gives a value between zero and one.

The second measure looks for evidence of noise in the window. Along each line, the number of transitions from light to dark and dark to light encountered in scanning along the line was counted, the values for the two lines were summed and the inverse of this quantity was taken as the value of the measure. If no transitions were observed, then the measure was set to one. For a perfect edge, as described above, this measure has a value of one. For a highly noisy edge, such as one in which there is no run of pixels of

constant value with length greater than one, the measure takes on a value close to zero.

Finally, a weighting factor was assigned to each of the measures, such that the weights summed to one. Each measure was multiplied by its weight and the results were added to give some estimate of the likelihood of an edge transition between the two lines of pixels. These estimates were computed for each pair of edges within the window and the maximum value over all such pairs was assigned to the window as a measure of the likelihood of an edge being present in that window.

In deciding on various parameters which went into the above algorithm, it became clear that interactive graphics would be useful. The above procedure was imbedded into a program which enabled the user to display the binary image on our Wicat graphics terminal, specify windows by hand, draw the best detected edge in the window, and print a hardcopy dot matrix display of the results. This package can be easily extended to enable rapid study of a variety of other binary edge processing algorithms.

Several windows were studied in the radar image to determine if the above approach was a fruitful direction for binary edge detection. Windows were placed in regions with no significant edges, windows with significant edges but not in the direction of the lines in the windows and windows with edges roughly parallel to the direction of the lines in the window. The experiments which will be described later in this section showed strong differences between the merits for windows with edges versus those without edges.

Two classes of comments should be made regarding the binary edge detection procedure described above. The first class of remarks deals with the problem of making the procedure computationally feasible. The second class of remarks deals with various extensions to the approach. In our application of binary

Figure 3-12b. Twice filtered and unfiltered radar image. Left-hand image in each set is the original radar image subwindow and right-hand image is the result of two iterations of edge preserving filtering.

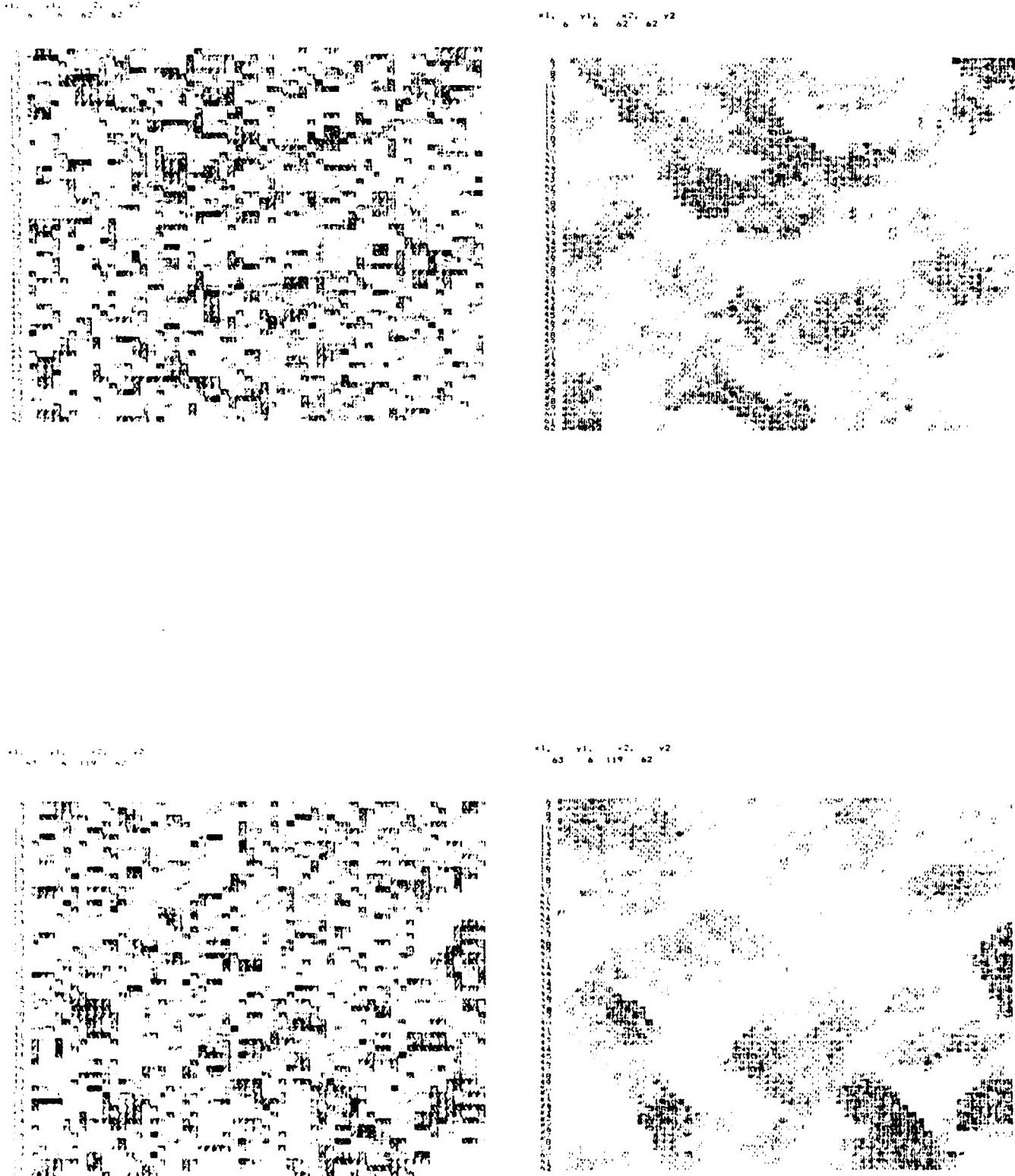


Figure 4-11. Radar window and results of edge preserving filtering
Application of image filtering algorithm (Lee 1981)
which uses edge information to decide on local window
shape and location to be used for filtering.



Figure 4-11a. Original radar window used in filtering.

overlapping 3x3 neighborhoods and the mean is computed for each. The nine means are then used to form a 3x3 window and a gradient mask is applied to the window. The value of the gradient in eight directions is determined using four gradient masks. That direction in which the gradient magnitude is largest is used to determine the direction of the line. Associated with each gradient direction is a subset of the 7x7 window to be used in the averaging process. The gradient directions are given in Figure 3-11(a) and the corresponding pixel subsets are given in Figure 3-11(b). The gradient masks are given in Figure 3-11(c).

Since the variance of the noise is not known, it is necessary to estimate this quantity. Over the full 7x7 window, one would expect much of the variation in observed pixel values to be due to the variation in the underlying real signal values.

As the size of the window shrinks, we might expect that most of the variation in observed pixel values is due to the variation in the noise. Roughly speaking, we are assuming that, on the average pixel values change slowly in regions without edges. The approach we used for noise variation estimation, was to compute the variance in each of the 3x3 sub-windows selected and take the average of the five smallest variances. Each variance of a 3x3 window is taken to be an estimate of the noise variance. Due to the small sample size, it is desirable to average several such variances to form a more robust estimate. Finally, the smallest variances are averaged since large variances are more likely to be due to variations in the underlying real pixel value.

The local filtering routine contains only one free parameter, the minimum value required in the 7x7 window for the window to be considered to have an edge and thus have the special neighborhood defined for filtering. A variety of values for this threshold were explored. We found a value of 50 give best

Figure 3-11. Information for edge preserving filtering.

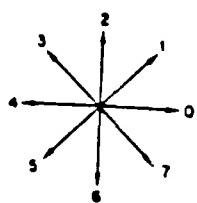


Figure 3-11a. (Lee 1981) Gradient directions.

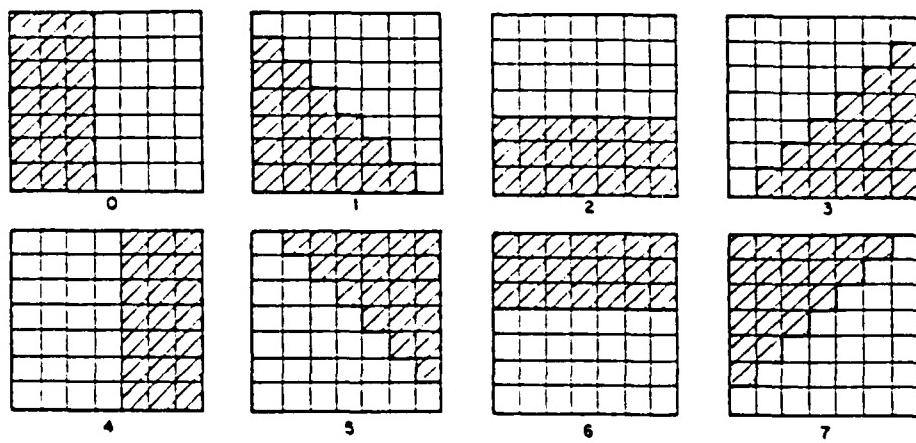


Figure 3-11b. (Lee 1981) Pixel subsets used in estimating correct pixel value for each of the eight gradient directions.

1	2	3	4
$\begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix}$	$\begin{matrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{matrix}$	$\begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix}$	$\begin{matrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{matrix}$

Figure 3-11c. Gradient masks for edge detection.

(i,j) . The a priori statistics are thus given by

$$xm(i,j) = zm(i,j),$$

$$Q(i,j) = E[(z(i,j) - zm(i,j))^2] - \sigma^2,$$

where zm denotes the mean of the observations $z(k,l)$ where (k,l) varies over the local window about (i,j) . In the above equations the sample mean and variance of z can be estimated from the local neighborhood of (i,j) . For the moment, it is assumed that the variance of the noise is known, but an estimation procedure will be described later.

In Lee [1980], the following minimum mean square filter is developed. The estimated value for the uncorrupted grey level of pixel (i,j) is denoted $xp(i,j)$. The optimal mean square estimate is

$$xp(i,j) = xm(i,j) + k(i,j)(z(i,j) - xm(i,j))$$

where

$$k(i,j) = Q(i,j) / (Q(i,j) + \sigma^2).$$

The weighting factor $k(i,j)$ lies between 0 and 1. Thus the estimated value of a pixel lies between its observed value and the mean of the pixels in its neighborhood. For low contrast areas, the estimate tends to lie close to the mean for the neighborhood and for noisy areas, the estimate is closer to the observed value.

If a square neighborhood of the point is used to estimate the required information, this may lead to problems. If an edge goes through the neighborhood, then the mean is computed using pixels on both sides of the edge which will tend to cause edges to be weakened. In our work, following Lee [1981], edge information is used to modify the shape of the neighborhood. First, the global variance in a 7×7 neighborhood is computed. If this variance is sufficiently large, this is taken as evidence of a possible edge in the neighborhood of the point. The 7×7 window is then divided into nine

3.4 Local Filtering

The difficulties in interpreting the radar image appear to stem, in part, from a high level of noise. A key criterion for a noise removal algorithm is the ability to remove noise from areas with little variation in the underlying signal, while preserving the contrast in edges. Extensive modeling of image statistics did not seem desirable given the limited data available. We instead sought an edge-preserving, noise smoothing procedure with an acceptable computational load which did not require extensive estimation of image characteristics. We selected an approach developed by Lee [1981] which attempts to use edge orientation in high contrast regions to define an appropriate neighborhood for estimation of the local mean and variance of the image. Lee claimed the algorithm preserved subtle details and lines while removing noise in flat areas and along edges. Our interest in this approach was increased by the fact that Lee's experiments were conducted on radar (Seasat) imagery.

We now describe the basic approach to estimating local image means and variances, the resulting minimum mean square error filter and the refinement developed by Lee which incorporates edge information to improve filtering.

An additive noise model is assumed for the image. Thus if we denote the observed (corrupted) brightness of pixel (i,j) by $z(i,j)$ and the pixel value of the uncorrupted signal by $x(i,j)$, then we assume the following relation holds:

$$z(i,j) = x(i,j) + w(i,j)$$

where $w(i,j)$ denote white noise with variance, σ^2 , and mean zero. The a priori mean, $xm(i,j)$ and the a priori variance $Q(i,j)$ of the uncorrupted picture are estimated using the local mean and variance of all pixels in a neighborhood of

Label	Edge	Maximum Merit
A	Yes	33 31 33
B	Yes	57 62 40 54 71
C	Yes	58
D	Yes	60 48 50
E	Yes	54
F	Yes	39
G	No	15
H	No	1
I	No	16

Maximum Merit		
	Average	Variance
Edge	47.7	10.1
No Edge	10.7	8.4

TABLE 3-1. Results from binary edge detection for radar image. Labels refer to parallelograms in Figure 3-10b.

the radar image. The binary edge detection appears promising as a means of overcoming the noise characteristics of the radar imagery. An alternate approach is to perform some form of edge preserving filtering. Work in this direction is described in the next section, Section 3.4.

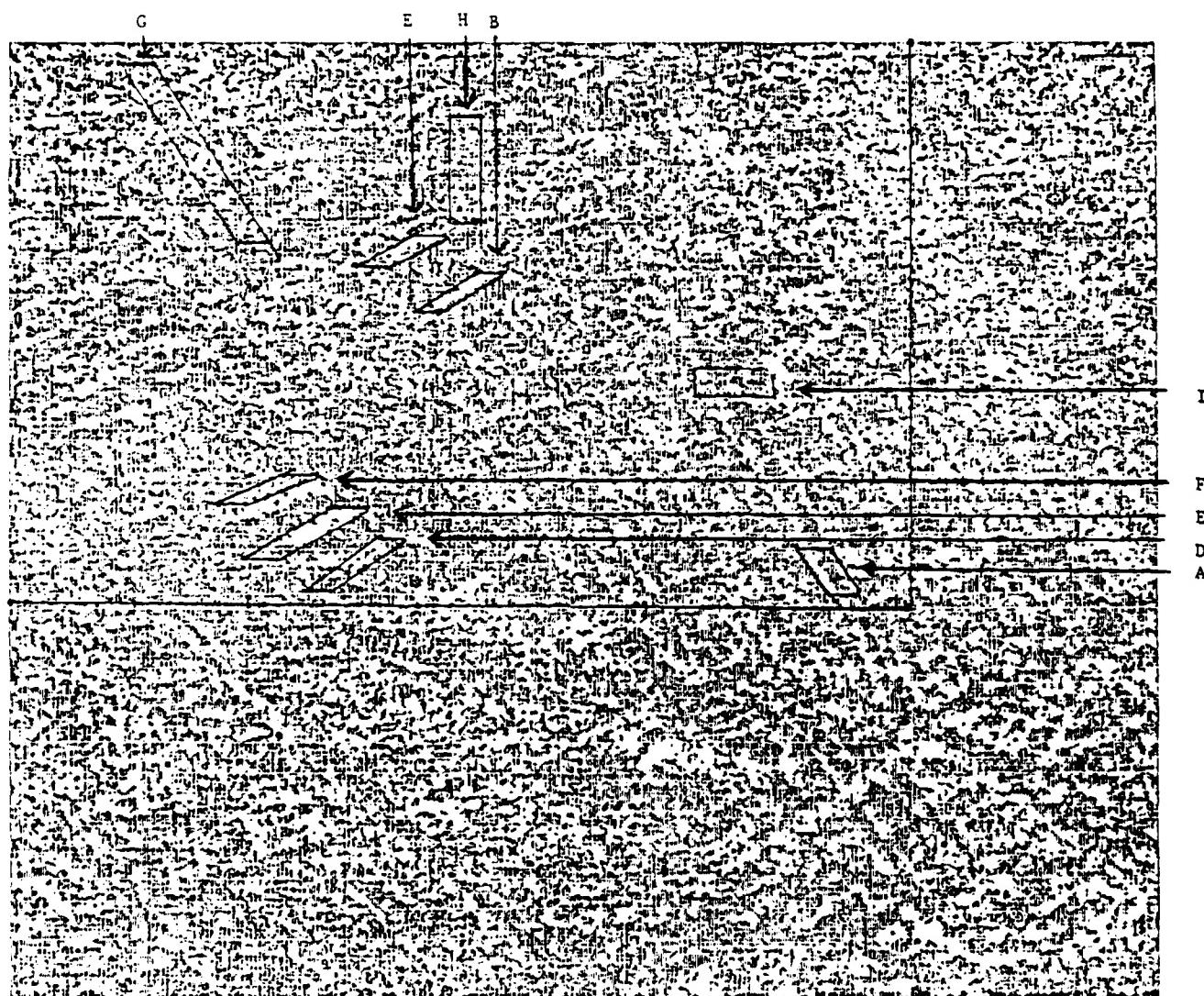


Figure 3-10b. Parallelograms for binary edge detection. Parallelograms A-F correspond to edges, G, H, and I do not.

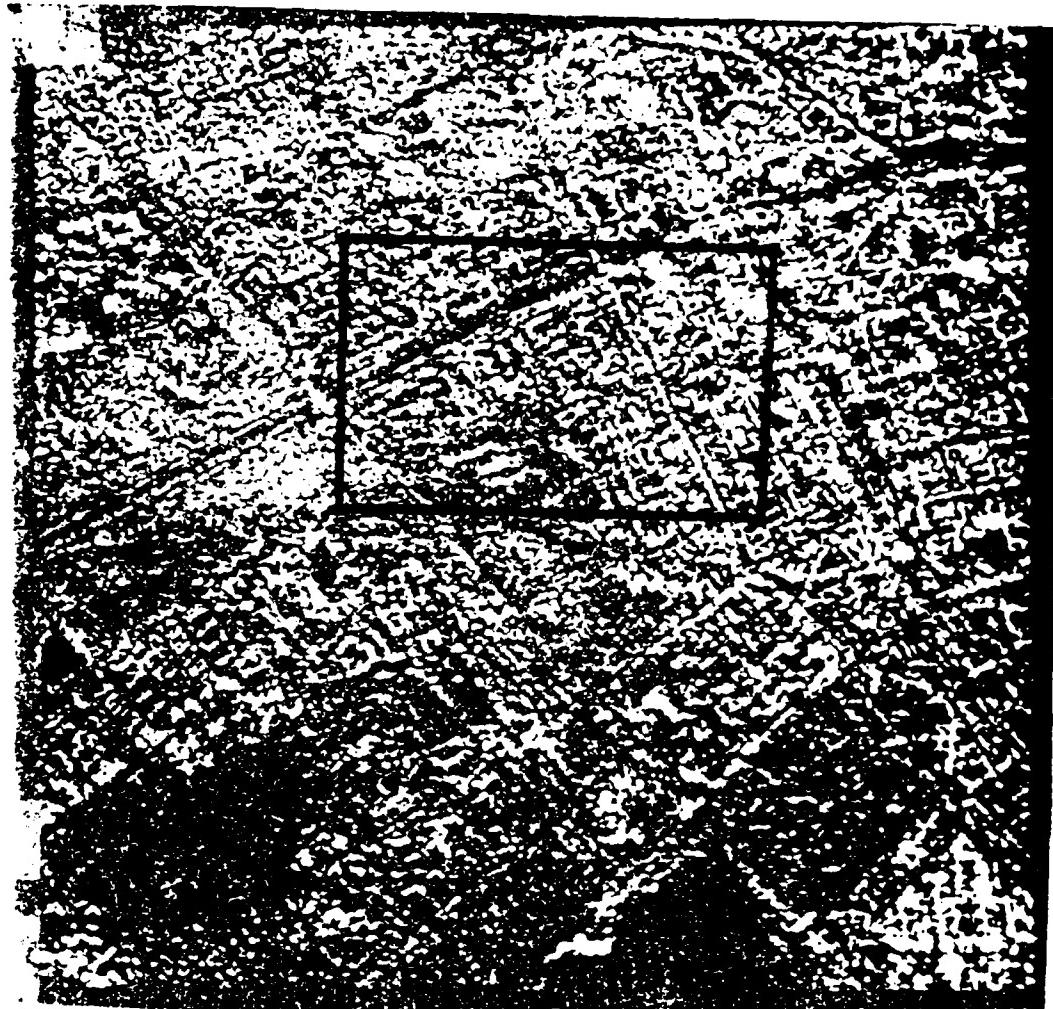


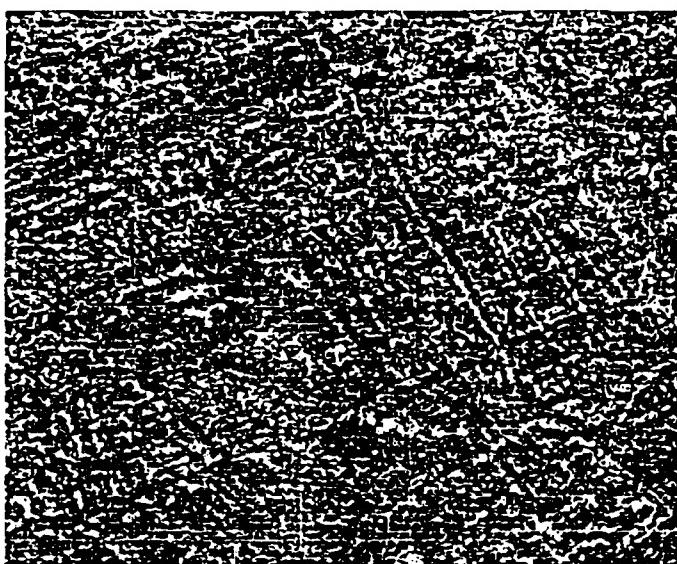
Figure 1-1(a). A 240 radar window used for shrinking and binary edge detection.

thresholding, a lower and upper thresholding is given. All pixels within that range are set to black and the remaining pixels are set to white. The grey level values are taken in the range 0-256. The results of thresholding in the range 10-80 is given in Figure 3-7 and in the range 20-60 is given in Figure 3-8. A third thresholding in the range 15-45 was performed. The dark areas were shrunk with one iteration of the shrinking algorithm using 4-neighbors. The results are shown in Figure 3-9. Of these the 20-60 case was selected as best showing the edge structure. The binary edge detection procedure was then applied to this image.

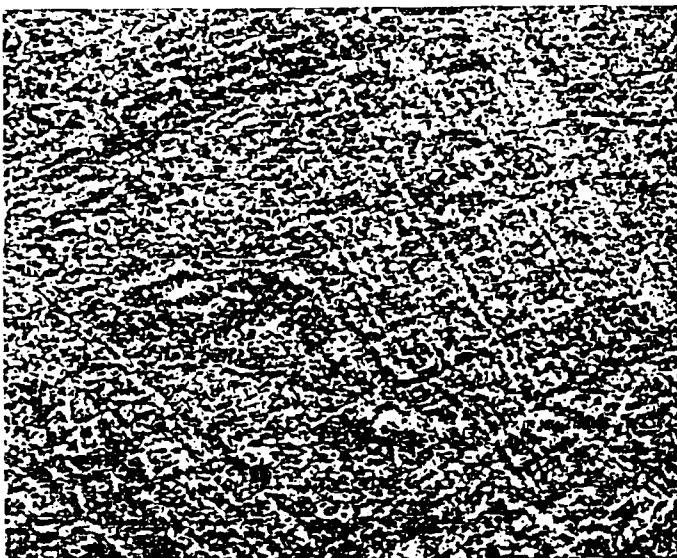
Our experiments with the binary edge detection approach used only the relative counts of black and white pixels on adjacent edges and not the measure based on the number of transitions. Six areas in the image which appeared to contain edges corresponding to roads were selected. For each of these areas a parallelogram with sides approximately parallel to the edge were placed over the corresponding area. The parallelogram was 11 lines wide (i.e. consisted of 11 parallel edges) along the direction parallel to the road. The parallelograms are displayed in Figure 3-10. On four of the edges, the location of the parallelogram was slightly perturbed to see the effect on edge detection. To each parallelogram, we can assign the maximum merit over all pairs of edges. The maximum merits for the parallelograms and their means and variances are displayed in Table 3-1. Based on these results, the approach bears further investigation. The mean merit difference between edges and non-edges is substantial, but experimentation should be performed with the algorithm extensions described above to get a better understanding of the usefulness of the algorithm.

The binary edge detection procedure described above is an attempt to overcome the poor performance of the registration programs' edge detection on

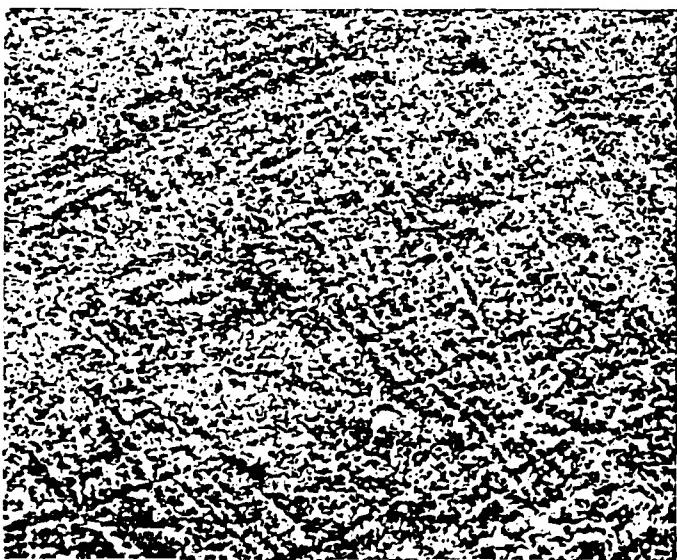
Figure 3-9. 512 X 512 radar image, thresholded with black region shrunk once using 4-neighbors.



(a) Black corresponds to pixel values 10 - 80.



(b) Black corresponds to pixel values 20 - 60.



(c) Black corresponds to pixel values 15 - 45.



Figure 3-7. Thresholded 512 X 512 radar window. Black areas are pixel values in range 10 - 80 out of 0 - 255.



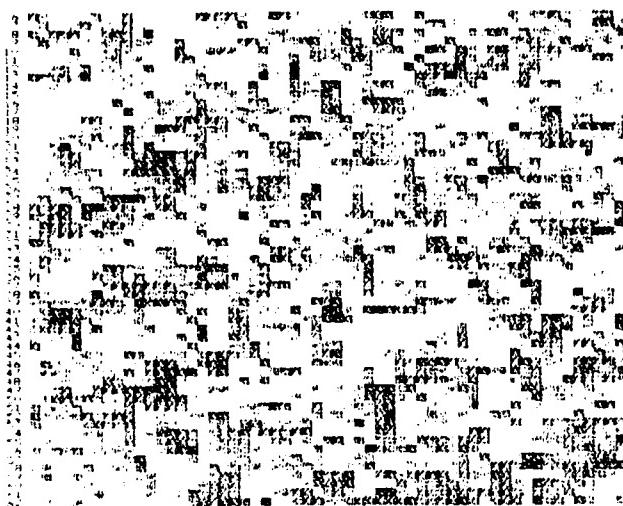
Figure 3-8. Thresholded 512 X 512 radar window. Black areas are pixel values in range 20 - 60 our of 0 - 255.

edge detection, windows were placed by hand. In a fully automatic approach, the windows could be placed in a large number of positions and orientations, but the amount of computation becomes formidable. Several approaches to this problem are possible. First the binary edge approach could be applied to verify the grey level edges found using the Hough transform in the registration program. Since the Hough transform performed poorly in locating images in the radar image, the binary approach could be exploited to provide an alternate means for verifying edges with error sources considerably different from that of gradient methods. The number of edges produced by the program on the radar image was sufficiently small that the binary approach could be efficiently used. A modified version of the Hough transform could also be performed directly on the binary image to get a rough indication of edge location and the above binary edge detection procedure could then be used for further edge verification.

Numerous extensions to the binary edge detection procedure are possible. The operation could be performed at varying resolutions to reduce some of the noise effect through resampling and to reduce the computational cost of the algorithm. More elaborate measures of edge continuity could be employed. For example, if several pixel value transitions are observed along a line, but they tend to clump near one end of the line, this may be stronger evidence of a homogeneous area such as a road than if the transitions are distributed along the length of the line. This could be incorporated into the binary edge procedure by adding a weighting factor to the measure based on pixel transitions which reflects the spatial distributions of the distributions and not just the count as is currently done.

The shrinking algorithm (using 4-neighbors) was applied to the 512x512 radar window. Several thresholds were used in the experimentation. In all

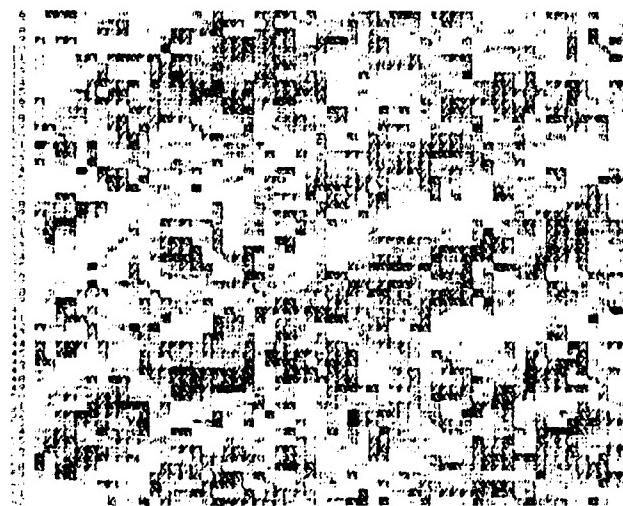
x1, y1, x2, y2
120 6 176 62



x1, y1, x2, y2
120 6 176 62



x1, y1, x2, y2
127 6 233 62



x1, y1, x2, y2
127 6 233 62

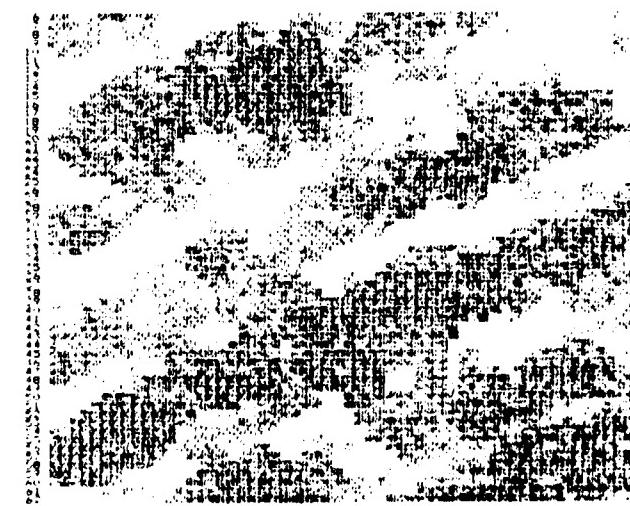
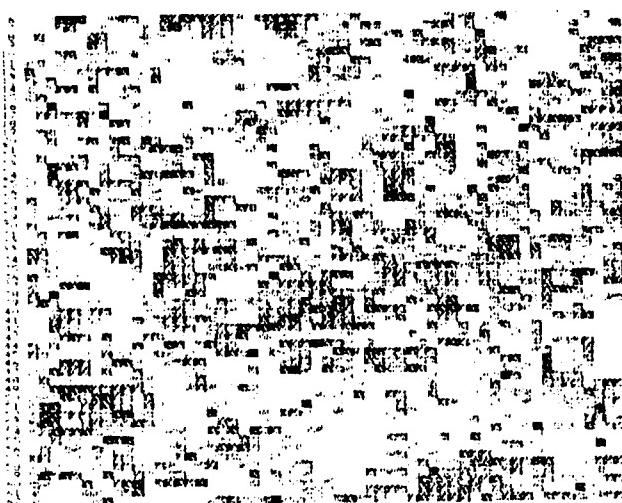


Figure 3-12b continued

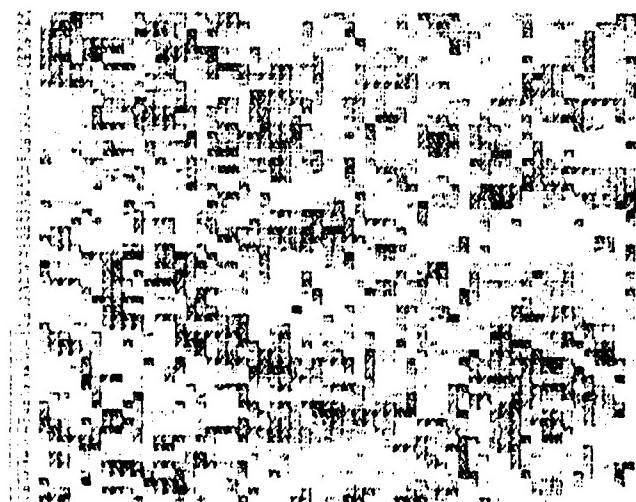
x1, y1, x2, y2
234 6 290 62



x1, y1, x2, y2
234 6 290 62



x1, y1, x2, y2
6 63 62 119

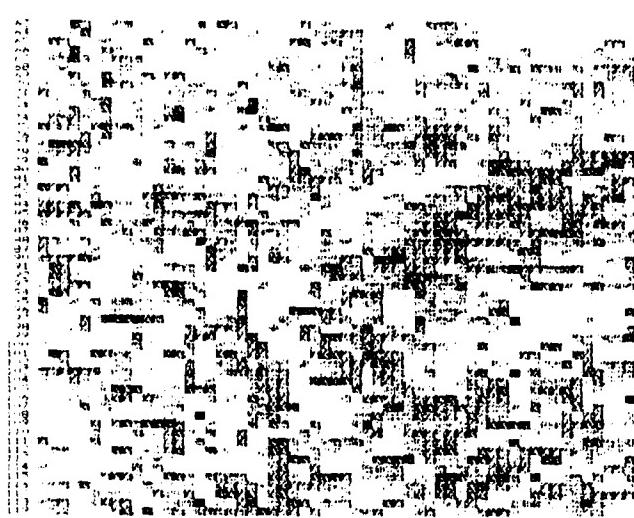


x1, y1, x2, y2
6 63 62 119



Figure 3-12b continued

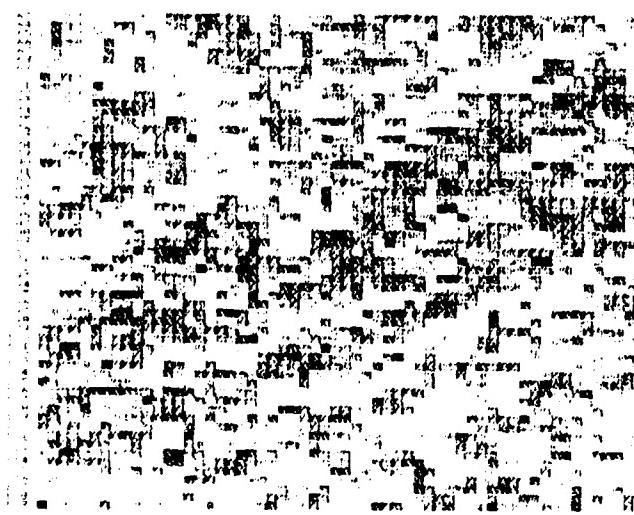
v1₁ v1₂ v2₁ v2₂
63 63 119 119



v1₁ v1₂ v2₁ v2₂
63 63 119 119



v1₁ v1₂ v2₁ v2₂
120 63 176 119

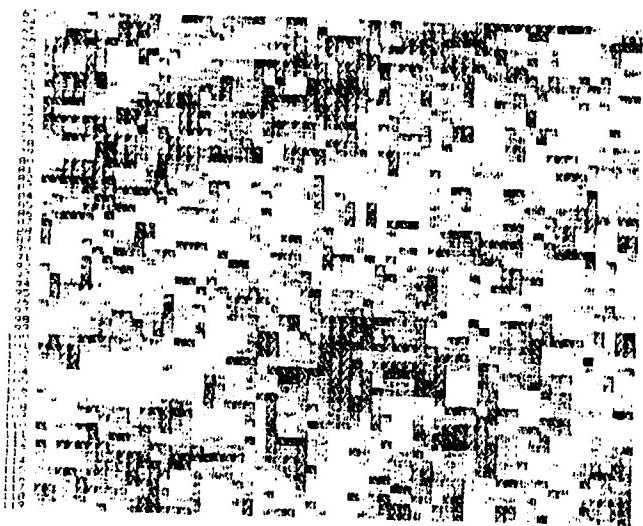


v1₁ v1₂ v2₁ v2₂
120 63 176 119

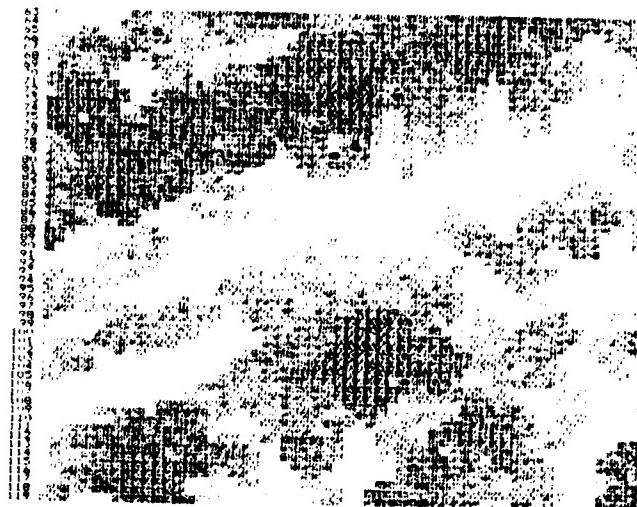


Figure 3-12b continued

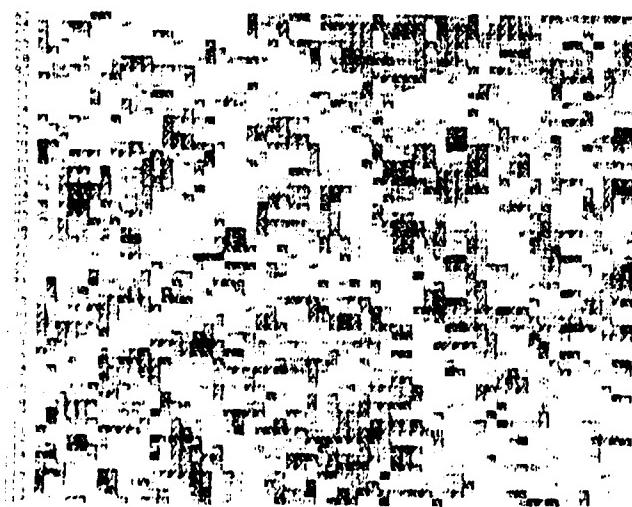
x1. y1. x2. y2.
177 47 233 119



x1. y1. x2. y2.
177 63 233 119



x1. y1. x2. y2.
234 63 290 119

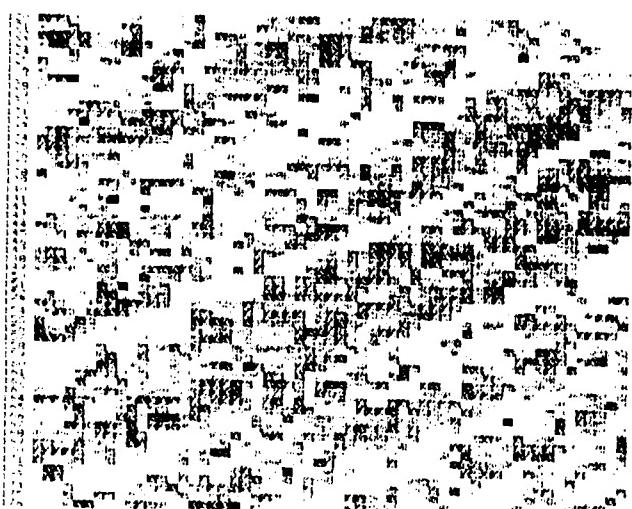


x1. y1. x2. y2.
234 63 290 119



Figure 3-12b continued

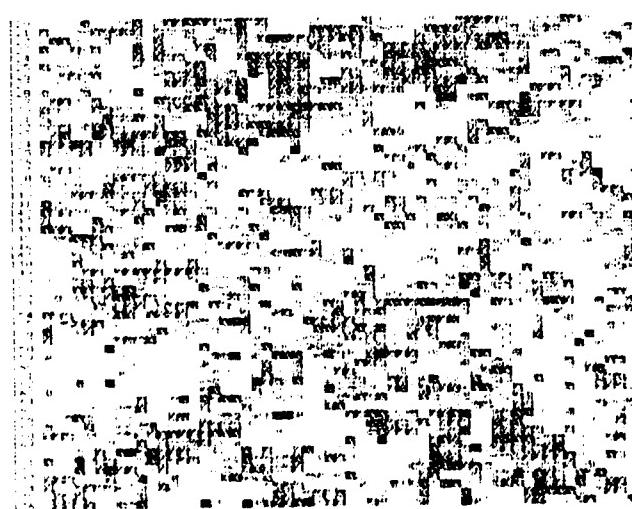
x1, y1, x2, y2
6 120 62 176



x1, y1, x2, y2
6 120 62 176



x1, y1, x2, y2
67 120 119 176



x1, y1, x2, y2
63 120 119 176

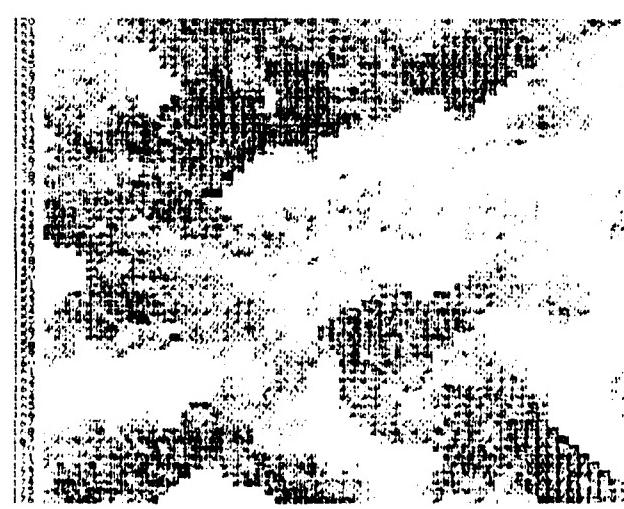
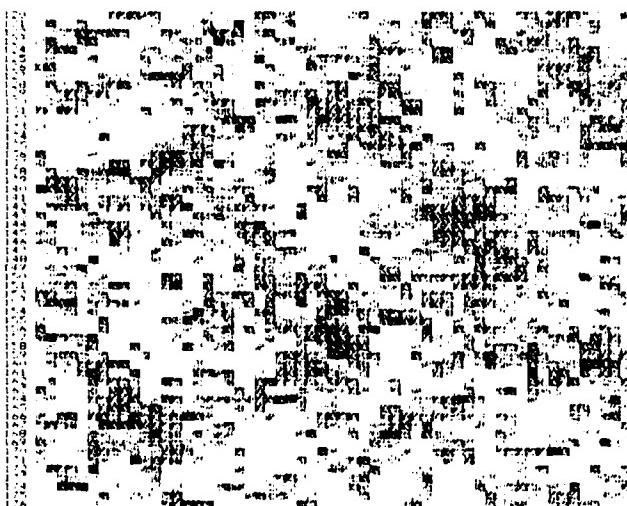


Figure 3-12b continued

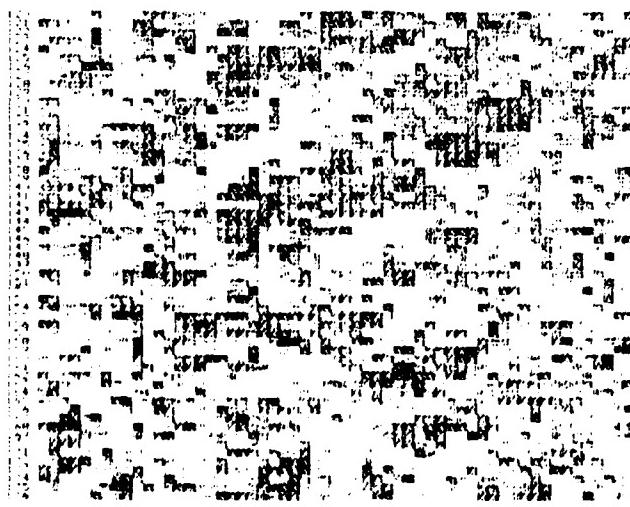
x1. v1. x2. v2.
120 120 176 176



x1. v1. x2. v2.
120 120 176 176



x1. v1. x2. v2.
120 120 233 176



x1. v1. x2. v2.
177 120 233 176

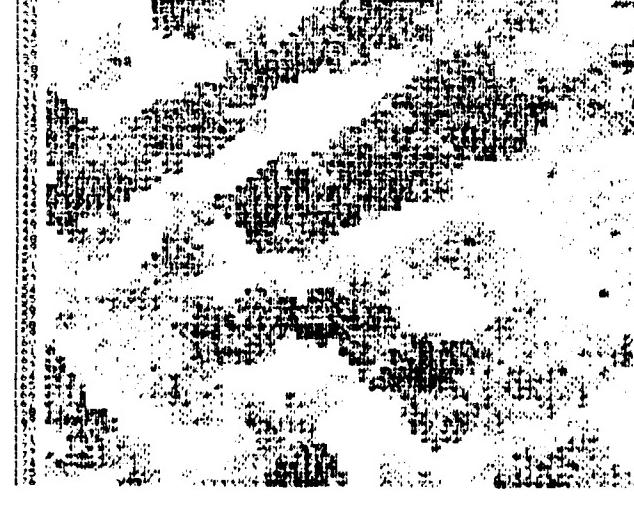
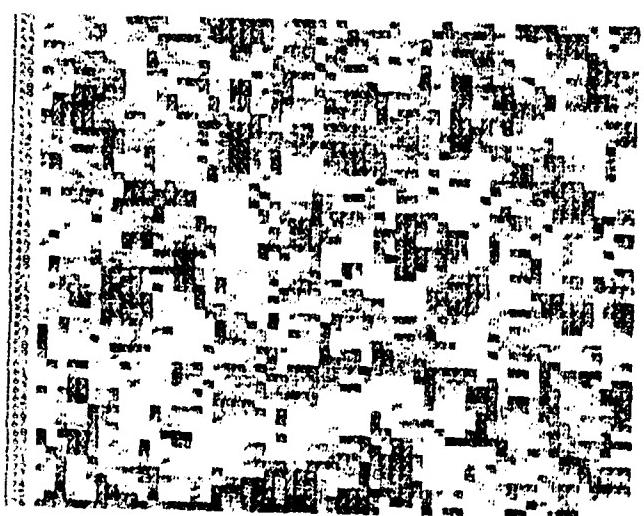


Figure 3-12b continued

x1, y1, x2, y2
234 120 290 176



x1, y1, x2, y2
234 120 290 176



x1, y1, x2, y2
6 177 62 227

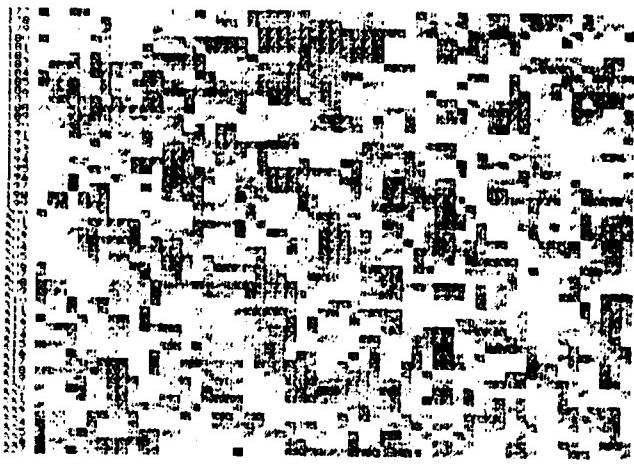


x1, y1, x2, y2
6 177 62 227



Figure 3-12b continued

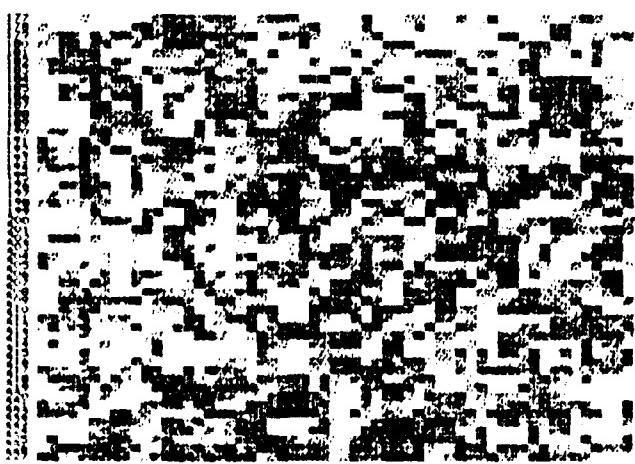
v1. v1. v2. v2.
63 177 119 227



v1. v1. v2. v2.
63 177 119 227



v1. v1. v2. v2.
120 177 176 227

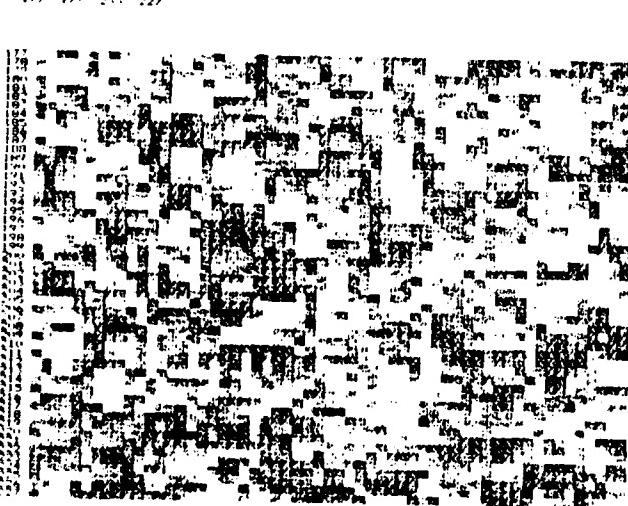


v1. v1. v2. v2.
120 177 176 227



Figure 3-12b continued

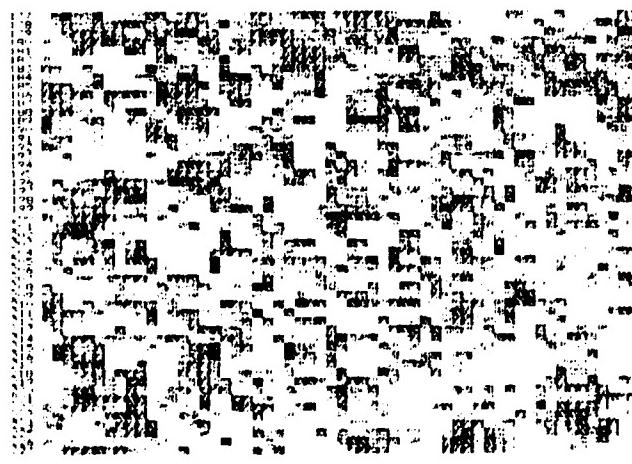
x_1 x_2 x_3 x_4 x_5 x_6



x_1 x_2 x_3 x_4 x_5 x_6



x_1 x_2 x_3 x_4 x_5 x_6



x_1 x_2 x_3 x_4 x_5 x_6



Figure 3-12b continued

results in terms of human interpretation. The results of two iterations of the filtering are shown in Figure 3-12. As can be seen there is considerable edge structure evident in the filtered image and much of the noise in non-edge areas has been suppressed. We considered the filtering process to be highly promising in bringing out the edge information required for intersection detection in the L.N.K. registration procedure.

3.5 Summary

The primary goal of the experimental work described in this section was to investigate methods for aiding in the extraction of features from radar imagery. This direction was pursued since the performance of the algorithm on optical imagery has been adequate in past experimentation, while radar imagery presents significant new problems. Several procedures for smoothing binary images, obtained from the radar image by thresholding, were applied to the radar image. The shrinking of regions using 4-neighbors was followed by the application of a binary edge detector. This approach may prove useful for verification of edges extracted using grey level information. An edge-preserving filtering algorithm [Lee 1981] was applied to the original radar image and resulted in considerable noise suppression, while enhancing the edge structure of the image. We consider this edge preserving filtering process to be extremely promising.

4.0 RIPS Conversion

L.N.K. was requested by the technical representative for the contract at ETL to help ETL personnel in putting up the registration procedure on their Cyber. This task was not part of the original statement of work for the contract and as a result, it was difficult to estimate the time required for the conversion. ETL provided an individual to actually implement the code on the Cyber, but L.N.K was responsible for providing ETL with a running version of the code in a version which could be readily converted to Fortran on the Cyber. In addition, L.N.K was to provide test data for the various routines together with sample runs of the program and documentation. Finally, L.N.K. was requested to help ETL with any problems relating to the working of the registration software while it was being put up on the Cyber.

The software conversion problem broke down into several subproblems. First, we faced the problem of what software to convert. The registration package existed on both the HP2100MX at ETL and on the Univac 1100 series. The HP version was designed to interact with the digitizer and display on the HP. L.N.K has not used this version for several years and the Univac version is much more sophisticated. One virtue of the HP system is its simplicity. The Univac version makes use of Fortran V features that would necessitate significant rewriting in adapting this code to run on the Cyber. In addition, the size of the Univac code was sufficiently large that L.N.K's role in the conversion process would have consumed a substantial part of the present contract. In light of the above considerations, it was decided, in agreement with ETL personnel, that the HP version of the registration software would be converted.

Difficulties in directly converting the HP version to run on the Cyber led

to a request by ETL for L.N.K to implement a version of the simpler registration software on the Univac 1100 series in a form that could be easily tested and converted to run on the Cyber. In response to this request, L.N.K implemented and tested a modified version of EDGEEX, a major module of the HP registration program, on the Univac. This software, together with details of its internal workings and test results were given to ETL and L.N.K provided further discussions to aid in the Cyber implementation of this code.

5.0 Probabilistic Modelling of Point Patterns

5.1 Introduction

The probabilistic modelling of point patterns is of considerable value in the analysis of image registration methods that are based on point matching. Analysis of the reliability of the results of such procedures is extremely important. Many registration procedures such as least squares fitting of control points and the L.N.K registration procedure return both an estimated transformation and a measure of the quality of the point fitting used to estimate this transformation. Using the measure of fit to estimate the reliability of the transformation is a complex problem. To illustrate a source of this complexity, consider a registration problem in which the model image consists of a set of points and the sensed image is obtained by translating the model image and then perturbing the coordinates of all points independently, using a uniform random number generator on the interval $[-a, a]$, $a > 0$, on each of the coordinates. The translation represents the actual offset between images and the perturbation represents measurement error in locating the points. The perturbation may result in the whole pattern being translated, in which case a perfect fit would be obtained using least squares, but the resulting transformation would be wrong. This would be the case, for example, if the perturbation for each point was a shift of $-a$ in the x direction and 0 in the y direction. The resulting shift estimation for x would be $-a$ using least squares or the L.N.K. procedure, however the true shift would be zero. In both cases a perfect match between model and image would be reported.

The example described above may seem quite artificial since the likelihood

between their orthogonal projection of these patterns onto T. Thus any point in the hypercube about p which has the same orthogonal projection onto T matches perfectly using the L.N.K. procedure. Thus if measurement error caused such a perturbed point to be produced, the L.N.K. procedure would result in a perfect match. Thus the procedure would estimate the translation between the model and the perturbed set and state that a perfect fit was found, while this transformation is actually in error.

If instead of perturbing p along the perpendicular projection of p onto T, we perturb it in a direction parallel to T, then the perturbed point, p' , will have the property that the distance between p and p' in R^4 is the same as the distance, in T, of the projections of p and p' onto T. Thus perturbations of p which are parallel to T give rise to errors in the transformation estimations and to a poor match, while perturbations along the projection of p onto T give rise to errors in the transformation but to a good match. If we know how common the latter situation arises, we can make use of a measure of pattern fit, such as the match weight produced by the L.N.K. registration algorithm, to determine how reliable the estimated transformation is.

The machinery has now been laid out to give one probabilistic formulation of registration error analysis. Let c be a threshold of pattern similarity required to consider two patterns to be a match. In the current registration program, this would be a value of the match weight between the sensed image and the model, which has to be exceeded if the resulting transformation estimate is to be accepted. For the present example, we replace the match weight by the distance between the projections of p and p' onto the pattern space T. The threshold, c, then defines a disk, D, in T whose center is the projection of p and whose radius is c. The only points in R^4 which will be considered to match with p are those with perpendicular projections onto T

confidence in the outcome of the procedure.

We consider the relatively simple case of a model image consisting of a set of points and a processed sensed image consisting of a translated version of the points in the model image followed by a perturbation of these points. Assume the perturbation of each coordinate of each point is done independently of every other coordinate. Further, assume that the value of a coordinate is perturbed by adding a random number chosen using a uniform distribution over the interval $(-a, a)$ for some $a > 0$. If the model pattern has n points and is represented by a point, p , in R^{2n} , then our perturbation procedure creates a new pattern which corresponds to selecting a point in a hypercube with side length a , centered on p , and with sides parallel to the coordinate axes. Thus the sensed pattern is a point p' which is obtained from p by perturbing p in its hypercube and translating the resulting perturbed point. For the rest of the present application, we will restrict our attention to the two-point pattern case described in Section 5.3. The general case of n -points is handled in a completely analogous manner, but the description is somewhat more complicated.

The random displacement of p within the hypercube can result in two sources of registration error. First the displaced point may match well using the L.N.K. registration procedure (restricted to the estimation of a translation), but the resulting transformation may be wrong. Consider the restricted case of a pattern of two points in the plane, where two sets of pairs of points are equivalent if they differ by a translation. In this case, which was considered above, it was determined that the space of patterns could be identified with a two-dimensional subspace, T , of R^4 , which was generated by the vectors u_3 and u_4 . Given two two-point patterns, their similarity as measured by the L.N.K. registration procedure is determined by the distance

identifying two points which differ by an interchange operator. Rather than deriving the resulting space, we observe that the space is easily obtained by lexicographical ordering of the points. Given two points $x=(x_1, y_1)$ and $y=(x_2, y_2)$ in the plane, we say that $x \leq y$ if

- 1) $x_1 < x_2$ or
- 2) $x_1 = x_2$ and $y_1 \leq y_2$.

Given a point $x=(x_1, \dots, x_{2n})$ in \mathbb{R}^{2n} , we say the point is lexicographically ordered if $2i+1 < 2j+1$ implies

$$(x(2i+1), x(2i+2)) < (x(2j+1), x(2j+2)).$$

Thus the pairs corresponding to points in the n-tuple are ordered from left to right. Geometrically, the resulting figure can be shown to be a cone in \mathbb{R}^{2n} .

5 Registration Reliability Example

We have described three pattern spaces, the space of patterns invariant under translation, patterns invariant under rotation, and patterns invariant under reordering of points. In our study, we were also able to develop a description of the space of patterns invariant under translation and reordering and the space of patterns invariant under rotation and reordering. We were unable, due to lack of time, to obtain an explicit description of the space invariant under all three operations, but based on our preliminary work on the problem, we think it is quite tractable.

We now consider an application of our work on identifying the space of point patterns. The first application deals with the problem of determining how likely it is that measurement error in locating the key point features such as intersections gives rise to high confidence in an incorrect transformation as opposed to giving low confidence in any transformation. This issue is of considerable interest in the analysis of the L.N.K. registration procedure performance because it provides some measure of

Note that in the translation case, the pattern space was an affine subspace of euclidean space. In the rotation case, the space requires a nonlinear transformation to turn it into a complex vector space.

The space of all patterns invariant under rotation is not a complex projective space since in the definition of complex projective space, it is assumed that the space is obtained by identifying certain sets of points on the unit sphere in complex n-space rather than in all of complex n-space. It is easily shown that rotating a set of points in the plane does not change the modulus of the corresponding number in complex n-space. Using this, it is possible to show that the pattern space for patterns invariant under rotation is essentially the cartesian product of complex projective space with a real line. Using the coordinate system described above on U_1 , it is possible to give an explicit coordinate system on the pattern space except on a subset which can be discarded without affecting our computations.

5.5 Reordering

The space of patterns invariant under a reordering of the points is simpler to describe than the other pattern spaces. Once again we view a set of n points as a point in $2n$ -dimensional euclidean space. Reordering the points merely corresponds to interchanging components in the corresponding n tuple in euclidean space. Thus if (x_1, x_2, x_3, x_4) is a point in R^4 , then this is equivalent to (x_3, x_4, x_1, x_2) since this corresponds to the same pair of points in the plane. Intuitively this means that we do not know which points in one pattern correspond to which points in another, so we wish to consider all possible orderings of the points to find the best match.

Interchanging components in an $2n$ -tuple is given by a linear operator on real $2n$ -space. Thus the space of all patterns invariant under a reordering of points is the same as the space of all points in real $2n$ -space after

euclidean space. The dimension of the euclidean space should not vary from point to point in the surface. Such a surface is called a real n-dimensional manifold. By replacing euclidean space with complex n-space and replacing differentiable maps with holomorphic (analytic) maps, we get an object called a complex n-dimensional manifold.

It can be shown that P^{n-1} is a complex n-dimensional manifold. We will not give a detailed proof of this but we will give a set of maps which can be shown to satisfy the requirements of a complex manifold. We will define a set of subsets U_1, \dots, U_n of P^{n-1} such that every point in P^{n-1} lies in at least one of the U_i and we will define one-to-one and onto analytic maps from each U_i to C^n . The U_i satisfy additional technical properties required to establish P^{n-1} as a manifold, but we refer to standard works [Chern 1979] for details. Let (z_1, \dots, z_n) denote a point in complex n-space. Denote the corresponding point in P^{n-1} by $[z_1, \dots, z_n]$. Define U_i by

$$U_i = \{[z] : z_i \neq 0\} \subset P^{n-1}.$$

Define h_i which maps U_i to C^n by:

$$\begin{aligned} h_i([z_1, \dots, z_n]) = & (z_1/z_i, \dots, z_{(i-1)}/z_i, \\ & z_{(i+1)}/z_i, \dots, z_n/z_i). \end{aligned}$$

The maps, h_i , are called coordinate maps since they provide a correspondence between U_i and C^n which has the standard coordinate system. Thus the only term which is omitted is z_i/z_i . We can now simplify matters considerably by noting that U_1 (or any other U_i) is defined on all of P^{n-1} except on a set defined by an equation involving one of the coordinates. It can be shown that this implies that the volume calculations required for our reliability estimates could be computed using the set U_1 instead of the set P^{n-1} without affecting the results. Since h_1 defines a 1-1 correspondence between U_1 and C^n , we can identify our space of patterns invariant under rotation with C^n .

material is available in [Chern 1979].

The essential idea in describing complex projective space is that it can be made to look locally like complex n -space and standard integration techniques on C^n can be applied to perform integration on projective space which we denote by P^{n-1} . This is essentially what is done in performing surface integrals in elementary calculus. There a function on a surface is "pulled back" to a function on the plane and the Jacobian is used to measure the local magnification in going from a small patch on the plane to a small patch on the surface. This is possible since a small patch on the surface looks like a small patch in the plane except that it must be flattened out.

This situation should be contrasted with the surface formed by taking the union of the xy -plane and the xz -plane in Euclidean 3-space. The neighborhood of any point on the x -axis in this surface looks like two intersecting surfaces rather than a patch on a surface. While integration over such a surface can be performed, it complicates the integration process. As long as a surface looks locally like a patch in the plane, we can pull back the integral of a function on the surface to an integral on the plane and then patch the integrals together. The sphere and the torus (surface of a doughnut) are two surfaces which have this property. This process of decomposing a surface into patches is somewhat like creating an atlas for the world. Each page in the atlas describes a region of the earth which can be flattened out to look planar and the atlas describes how the different charts overlap.

The basic property we impose on surfaces is that each point have a neighborhood (the set of all points in the surface within a specified distance from the given point) such that there is an infinitely differentiable function, with infinitely differentiable inverse, from the neighbor onto

of modulus one. Hence the space of point patterns where two patterns are regarded as being the same if the patterns differ only by a rotation, can be regarded as the space obtained from C by identifying two points (z_1, \dots, z_n) and (w_1, \dots, w_n) if there exists a complex number, u , such that $(z_1, \dots, z_n) = (uw_1, \dots, uw_n)$.

We now arrive at a somewhat delicate stage in the description of the space of point patterns invariant under rotation. The object obtained by making the identification described above is related to a well known entity, used in the fields of differential geometry and several complex variables, called complex projective space. Since our goal is to define an intrinsic space of patterns invariant under rotation, translation, and reordering of the points, then it will be necessary to make further identifications on this space before we arrive at the desired space. In order to compute our estimates of the registration algorithm reliability, the pattern space should have a notion of distance defined on it, where distance in this space should correspond to some intuitive notion about the similarity between patterns. Furthermore, the resulting space should have a notion of volume which is related in a natural way to the notion of distance. This enables us to talk about the size of the set of patterns which are similar to a given pattern. Details of this matter are involved and are not required for the pure translation problem described above, since the space is a vector space and ordinary integration in euclidean space is applicable. It can be shown that integration can be defined in a natural way on complex projective space. Since we must make the additional identifications for translation and reordering of points, it is important to keep track of the structure of the spaces we get after each of the identifications. We include some background necessary to understand the structure of the spaces, but forgo complete rigor. A treatment of this

$y=((t_1, u_1), \dots, (t_n, u_n))$ be two n -point patterns, where $r_i, t_i, i=1, \dots, n$ represent radial coordinates and $s_i, u_i, i=1, \dots, n$ represent angular coordinates. Then we say that x and y are equivalent under rotation if there exists a real number w such that

$$s_i = u_i + w \pmod{2\pi}, \quad i=1, \dots, n.$$

Ignoring the origin, we may think of polar coordinates as naturally sitting on a half-infinite cylinder in which the radial coordinate is zero at one end of the cylinder and goes to infinity as we travel along the axis of the cylinder and the angular coordinate varies from zero to 2π as we go around the cylinder. A set of n points may be represented as the cartesian product of n such cylinders. We then wish to identify points in this product if they differ by the above relation. While this formulation of the problem is relatively easy to motivate, we found it difficult to pursue. A more profitable approach is to use some basic results from the theory of n -complex variables.

Instead of viewing an n -point pattern as being a point in a $2n$ -dimensional real vector space, we treat it as a point in C^n , the cartesian product of the complex plane with itself n times. The advantage of this formulation is that rotation about the origin corresponds to multiplication by a complex number of modulus one. Let $X=[(x_1, y_1), \dots, (x_n, y_n)]$ denote a set of n points. Define n complex numbers, z_1, \dots, z_n , by $z_j = x_j + i(y_j)$, where $i = \sqrt{-1}$. Then the point set, X is represented by the n -tuple (z_1, \dots, z_n) . Recall that a complex number, $x+iy$, is of modulus one if $x^2 + y^2 = 1$. Multiplication of a complex number $w=u+iv$ by a complex number $z=x+iy$ of modulus one corresponds to rotating the point (u, v) , in the complex plane, about the origin. Thus rotating the points in X about the origin corresponds to multiplying each of the numbers z_1, \dots, z_n in the corresponding n -tuple by the same complex number

determines that translation which minimizes the sum of squares of the lengths of the shifts required to make the points align exactly, given that the translation has been performed. The distance in the point pattern space is related to the amount the points have to be moved to make the set of interpoint distances match. Since the registration program attempts to match using interpoint distances and directions, the notion of distance in the space of point patterns appears more natural than the least squares distance. The above remarks can be made precise, but the algebra is somewhat messy.

Another fundamental difference between our approach and least squares is that our approach is designed for analysis of registration procedures on point patterns and not for estimation of the translation between patterns. Thus the intrinsic pattern space approach avoids estimation of the offset and directly determines the distance between the patterns, by using the Euclidean distance on the pattern space. Least squares involves the solution of a system of equations, which are often nonlinear if operations other than translation are considered, whereas the intrinsic pattern space approach provides closed form formulas for the distance (similarity) between patterns. The comparison of the intrinsic pattern space with least squares is only tangential to the present discussion, since our goal is to define a pattern space on which to place a probability measure and least squares is not related to that issue.

5.4 Rotation

The space of point patterns invariant under translation has been explicitly described as a vector space and procedures for taking sets of points and finding their representatives in this space have been discussed. We now turn to a related, but more difficult problem, that of describing the space of point patterns invariant under rotation. The discussion is simplified if we use polar coordinates. Let $x=((r_1,s_1),\dots,(r_n,s_n))$ and

Gram-Schmidt procedure from classical linear algebra is a standard approach to doing this, but, in the present case it is easy to see that if we define $u_3 = (1/\sqrt{2}, 0, -1/\sqrt{2}, 0)$ and $u_4 = (0, 1/\sqrt{2}, 0, -1/\sqrt{2})$, then u_1 , u_2 , u_3 , and u_4 form an orthogonal basis for \mathbb{R}^4 and u_3 and u_4 form an orthonormal basis for the pattern space of interest.

Define T to be the space spanned by u_3 and u_4 . Then T represents the pattern space of all two-point patterns. Given any representative, $x = (x_1, x_2, x_3, x_4)$ of a pattern, we obtain its canonical representation as a point in T by taking the orthogonal projection of x on the subspace T . The projection, $p(x)$, of x is given by:

$$p(x) = (x \cdot u_3)u_3 + (x \cdot u_4)u_4.$$

More explicitly, we have

$$\begin{aligned} p(x) &= ((x_1, x_2, x_3, x_4) \cdot (1/\sqrt{2}, 0, -1/\sqrt{2}, 0))(1/\sqrt{2}, 0, -1/\sqrt{2}, 0) \\ &\quad + ((x_1, x_2, x_3, x_4) \cdot (0, 1/\sqrt{2}, 0, -1/\sqrt{2}))(0, 1/\sqrt{2}, 0, -1/\sqrt{2}) \\ &= 0.5(x_1-x_3, x_2-x_4, x_3-x_1, x_4-x_2) \\ &= 1/\sqrt{2}((x_1-x_3)u_3 + 1/\sqrt{2}(x_2-x_4)u_4) \end{aligned}$$

Thus the coordinates of $p(x)$, viewed as a point in T and with respect to the basis u_3 , u_4 are $1/\sqrt{2}((x_1-x_3), (x_2-x_4))$. Up to a scalar factor which is the same for all projections, this is just the difference vector for the two points, which agrees with our intuitive understanding of the situation. This approach is easily extended to the case of n -point patterns since the only operations required were the extension of a set of linearly independent vectors to form a basis and orthogonal projection, both of which are easy to perform for arbitrary vector spaces.

The relationship of the space of point patterns invariant under translation with the least squares approach to the estimation of the translation offset between two point sets requires comment. Least squares

patterns have the same difference vector.

We now illustrate the formal approach to deriving the intrinsic space of two-point patterns. The set of all pairs of points in the plane can be identified with R^4 , the four dimensional Euclidean space. Let $x=(x_1, x_2, x_3, x_4)$ be a two-point pattern in R^4 . We think of (x_1, x_2) and (x_3, x_4) as representing two points in the plane. We wish to identify two patterns $x=(x_1, x_2, x_3, x_4)$ and $y=(y_1, y_2, y_3, y_4)$ if there exists real numbers a and b such that $(x_1, x_2, x_3, x_4) = (y_1+a, y_2+b, y_3+a, y_4+b)$. Thus we identify two points, x and y , in R^4 if the pair of points in the plane which correspond to x can be obtained from the pair of points in the plane which correspond to y by a translation in the plane.

An arbitrary translation of a pair of points in the plane corresponds to adding a linear combination of the vectors $u_1=(1, 0, 1, 0)$ and $u_2=(0, 1, 0, 1)$ to the point in R^4 corresponding to the pair of planar points. For example the translation (a, b) can be written as

$$a(1, 0, 1, 0) + b(0, 1, 0, 1).$$

Given any point (x_1, x_2, x_3, x_4) , the set of all points equivalent to it under translation is that two-dimensional subspace of R^4 which is generated by u_1 and u_2 and which passes through the point (x_1, x_2, x_3, x_4) .

If we identify the set of equivalent points to a single point, the resulting pattern space can be represented as the two-dimensional subspace of R^4 which is perpendicular to the space generated by u_1 and u_2 . Recall in the simple example given in Section 5.2, each horizontal line represented a set of equivalent points which were identified to a single point on the y -axis. The y -axis was the resulting pattern space.

A basis for this new space can be obtained by extending the set of vectors containing u_1 and u_2 to give an orthogonal basis for R^4 . The

work, such as more expensive feature extraction, is required to obtain a translation estimation which meets accuracy requirements. Many mathematical details of our work on reliability estimates have been suppressed from the following exposition, since they would greatly lengthen the work. A more formal presentation of this work is being prepared for publication.

5.3 Translation

The problem of defining an intrinsic representation of the space of point patterns may be viewed as the problem in removing redundancy in representing points patterns as sets of points in the plane. To illustrate this idea consider the set of all point patterns in which there are exactly two points in each pattern, and call these patterns two-point patterns. Assume that the observer sees a translated but not rotated version of the true set of points. Using the notation above, let A and B be two-point patterns, where $A=[a_1, a_2]$, $B=[b_1, b_2]$, $a_i=(a_{xi}, a_{yi})$, $b_i=(b_{xi}, b_{yi})$, where a_{xi} , a_{yi} , b_{xi} , and b_{yi} are real numbers. Then A and B represent the same pattern if there exist a translation (x_t, y_t) , with x_t and y_t real numbers, such that

$$a_{xi} = b_{xi} + x_t, \text{ and}$$

$$a_{yi} = b_{yi} + y_t.$$

We now develop an explicit description of the space of two-point patterns as an illustration of the general approach to the development of an intrinsic description of the space of point patterns. This case has the advantage that it is not hard to guess the result. Given two points, we can represent their pattern by the difference vector between them. Thus, two points (x_1, y_1) and (x_2, y_2) can be represented by the vector $(x_2 - x_1, y_2 - y_1)$. Intuitively the difference vector captures all relevant information about the pattern given that we identify two patterns which differ by a translation, since two such

equivalent. Note that if one pattern is equivalent to a second and a second is equivalent to a third, then the first is equivalent to the third. Using this, it is easy to see that the set of all points in \mathbb{R}^{2n} can be partitioned into disjoint sets such that two points, each of which represent one pattern, belong to the same set if and only if they are equivalent. We then define a new set, say B , which contains one point for each set in the partition of \mathbb{R}^{2n} . We say that B is obtained from \mathbb{R}^{2n} by identifying points under the equivalence relation defined by translation. The set B then may be thought of as the set of all point patterns invariant under translation. We would like to endow B with a distance measure such that the distance between two points in B is a reasonable measure of the similarity of the corresponding patterns.

We will work out an explicit description of the space of point patterns latter, but we first give a simple illustration of the operation of identifying points under an equivalence relation. Consider an equivalence relation defined on the plane by saying that two points are equivalent if they lie on the same horizontal line. In this case, the y -axis is a specific example of an identification space under the equivalence relation, since we can assign to each horizontal line (a set of equivalent points) the point on the intersection of this line with the y -axis. In this case, the distance on the y -axis corresponds to the distance between the corresponding lines, and this distance is the distance measure that would be used in this space.

In the remainder of this section, we will develop explicit descriptions of several important spaces of point patterns and illustrate, in the case of translations, how this space can be used in estimating the reliability of an estimated translation given a measure of how well the corresponding point sets match. This reliability estimate could then be used to decide whether further

of a perturbation causing a pure translation seems small. Many perturbations, however, may result in a subset of the points being approximately translated. When we consider rotations and uncertainty as to which model points correspond to which image points, the likelihood of perturbations resulting in seemingly good matches, but incorrect transformations increases. Thus the problem of estimating the reliability of a transformation is not simple because perturbations may mask as transformations. In this section, we develop an approach to this problem based on the idea of a space of point patterns with respect to a class of transformations.

5.2 Pattern Spaces

Consider a class of point pattern registration problems with respect to a collection of transformations. If we are interested in estimating a translation offset between model and sensed image and we know which points correspond, then we want to consider two sets of points as being the same if they differ only by a translation. If we are interested in estimating a rotation and translation and we know which points correspond, then we consider two sets of points as being the same if they differ only by a translation and a rotation. If we do not know which points correspond to which in the latter case, then we consider two sets of points as being the same if there is some one-to-one correspondence between the points in one set and the points in the other such that under this correspondence, the points differ by a rotation and a translation.

Our basic approach to defining the space of point patterns is to first observe that a set of n points in the plane can be considered as a point in \mathbb{R}^{2n} , simply by putting the $2n$ coordinates of the n points into a list. Assume that we consider two patterns of n points which differ by a translation to be

lying within D . This set is a cylinder in R which is the cartesian product of a disk with a plane. Since our perturbations are restricted to lie in the hypercube defined previously, all matching patterns will lie in the intersection of this cylinder with the hypercube. Call this intersection H .

Our ultimate interest is in the accuracy of the registration transformation so it is natural to determine the probability that the estimated transformation is off by more than a specified amount, d , given that the merit of pattern match is at least the threshold c . Recall that the set of all translates of the planar point set corresponding to p is exactly the set $p+U$, where U is the two dimensional vector space spanned by the vectors u_1 and u_2 described previously. Let p'' be the point in $p+U$ which is closest to p' in R^4 . Given a registration procedure which finds the translation minimizing the sum of the squares of the interpoint distances between the translated model and the sensed image, it can be shown that the translation estimated by the registration procedure is $p''-p$. Let q be the point in $p+U$ which represents the actual translation of p . If there were no perturbations, only a translation, then we would have $p'=p''=q$. In any case, the correct translation is given by $q-p$. One measure of the error in the translation estimate is then given by the magnitude of the difference between the correct translation, $q-p$, and the estimated translation, $p''-p$. Thus a measure of the error in the translation is given by $|q-p''|$.

Now consider the set, M , of all points r in H (the set of perturbations which give rise to a pattern close to the projection of p in T), such that the projection of $r+q$ onto $p+U$ lies within a distance d , in $p+U$, of q . Any point in M will satisfy the condition that the estimated error in translation is less than the specified threshold d . Thus the probability that the estimated translation error is less than d , given that the measure of pattern similarity

(as measured by the distance of the corresponding patterns in T) is less than c , is given by the ratio of the volume of M to the volume of H . The volumes to be computed have boundaries which are given by either linear or quadratic expressions.

5.7 Summary

The computation of the volumes required for the translation error computations appears feasible, but since we were ultimately interested in the corresponding result for the case where rotation, translation, and reordering of points is considered, we thought it more reasonable to focus attention on that case. The above analysis applies to a procedure which finds a translation minimizing a function of the set of all interpoint distances instead of the distances between corresponding points as does least squares. The L.N.K. procedure works with subsets of the interpoint distances and their directions. Further work will be required to clarify this relationship.

In addition to the problem of adding rotations and point reordering to the translation case, we must also consider the fact that the number of points in the sensed image will be different from the number in the model. Our basic approach to this extension is as follows. The coordinates of a point in the pattern space, under any of our models, represents a relationship between points in the original planar point pattern. In the case of our two-point planar pattern model, the coordinates of a point in pattern space are the coordinates of the difference vector between the two points in the plane. For simplicity, assume that our set of model points, suitably translated and perturbed, is a subset of our sensed image points. If we take our pattern space to be defined using the number of points in the sensed image, then the model will no longer correspond to a point in pattern space, but to a whole

subset. This subset is defined by letting the coordinates corresponding to points in the sensed image but not in the model be allowed to vary arbitrarily.

The distance between a model and a sensed pattern is now represented by the minimum distance from the sensed pattern to the subset of pattern space corresponding to the model. We have only begun to work out the details of this extension but it has a great deal of appeal in that it lends itself to analysis of matching methods which try to maximize the number of good fits, as does the L.N.K. registration procedure, rather than trying to minimize least squares error which is subject to problems with outliers.

The fundamental purpose of developing an explicit description of the space of all point patterns invariant under rotation, translation, and reordering of points was to obtain a space on which to define a probability measure and to use this together with the information on how the pattern space is formed from the original Euclidean or complex space for the analysis of our registration program. This would enable us to estimate the probability that very dissimilar patterns would be matched using the registration program. It would also enable us to determine how far, on the average, a pattern could be disturbed before it would fail to match with itself. Furthermore, we could determine the significance of a measure of pattern similarity, such as the match weight used by the L.N.K. algorithm, in determining the reliability of the resulting transformation. Since reasonable error analysis would be difficult to carry out without an explicit description of the pattern space, we dedicated a considerable effort to its determination.

We think this work can be the base for a realistic theory of point pattern matching relevant to the analysis of registration algorithms using point matching in which rotation and translation must be estimated. While

related problems have been studied [Santalo 1976], they could not be directly applied. In addition to completing the work on the space of patterns invariant under all three operations, the extension of this work to include perspective transformations would be of considerable interest since it adds another level of realism to the problem. We think it likely that the above methods can be extended to the perspective case, but our examination of this situation was very brief.

6.0 Conclusions and Future Work

Under the present contract, we have developed some important mathematical tools for use in the analysis of algorithms which use patterns based on points such as the L.N.K registration algorithm. This work is a step towards more reliable modelling of feature patterns. A definition of point patterns is developed and various spaces of point patterns are defined which provide the natural settings for probabilistic analysis of the accuracy of the point matching in the L.N.K registration procedure.

The feasibility of registering optical imagery with radar imagery was explored. Using hand-picked points from both the optical and radar imagery, the registration algorithm was able to find the proper transformation in 22 out of 24 trials. The various trials included varying amount of noise points. This result is quite encouraging, in view of the considerable perspective distortion in the radar imagery, which was not modelled. Including perspective in the registration algorithm should increase the reliability of the procedure further.

Algorithms for use in the extraction of features from a radar image were investigated. A procedure for extracting edges in thresholded images was developed and some experimentation was performed on radar imagery. An image filtering algorithm which uses edge information to guide the filtering process was implemented and applied to the radar imagery. These approaches appear promising for incorporation into the general registration procedure to handle radar data.

There are several directions for future research which we feel would considerably enhance the usefulness of the L.N.K registration algorithm:

- 1) Completion of the description of the space of point

- patterns invariant under rotation, translation, and
reordering of points,
- 2) Extension of the concept of pattern space to include
perspective distortion,
- 3) Probabilistic analysis of the registration procedure
using the space of patterns,
- 4) Numerical computation of translation estimation
reliability,
- 5) Refinement of the binary edge extraction procedure to
achieve reasonable computational speed and accuracy,
and
- 6) Feature extraction work on images obtained using the
edge-preserving filtering process.

7.0 References

- S. Chern, Complex Manifolds Without Potential Theory, Springer-Verlag, New York, 1979.
- B. Lambird, D. Lavine, G. Stockman, K. Hayes, and L. Kanal, Study of Digital Matching of Dissimilar Images, ETL-0248, USAETL, Fort Belvoir, VA, 1981.
- D. Lavine, B.A. Lambird, L.N. Kanal, Analysis and Simulation of Discrete Digital Image Matching, ETL-0278. Final Report, USAETL, Fort Belvoir, VA, 1981.
- J. Lee, "Refined filtering of image noise using local statistics," Computer Graphics and Image Processing, Vol 15, 1981, pp.380-389.
- D. Marr and E. Hildreth, "Theory of Edge Detection," A.I.M. 518, M.I.T., April 1979.
- W. Newman and R. Sproull, Principles of Interactive Computer Graphics, McGraw-Hill, New York, 1973.
- A. Rosenfeld, and A.C. Kak, Digital Picture Processing, Academic Press, New York, 1976.
- L.A. Santalo, Integral Geometry and Geometric Probability, Addison-Wesley, Reading, Mass., 1976.
- J. Serra, Mathematical Morphology, Academic Press, New York, 1982.
- G. Stockman, B. Lambird, D. Lavine, L. Kanal, Knowledge-Based Image Analysis, ETL-0258, Final Report, USAETL, Fort Belvoir, VA, 1981.

1.0 Image Processing Library

This Appendix contains a brief description of the basic image processing functions developed by L.N.K. Corporation. This development was funded in part by this contract. The library was implemented on the super microcomputer, the WICAT 156, which is based on the 68000 microprocessor. A frame grabber, capable of digitizing an analog video signal, storing one video frame, and displaying the digitized frame on a monitor, has been added to the WICAT. The resolution of the frame grabber is presently 320 pixels horizontal by 240 pixels vertical. Sixty-four grey levels of intensity are provided. The frame grabber was manufactured by Datacube, and is referred to as the Datacube within the documentation.

Appendix A

```
-----  
| Parameters          AVEPROC -- AVERAGES GREY LEVELS <LNK .MRL>  
| LC      integer    -Left column of desired window  
| RC      integer    -Right column of desired window  
| TR      integer    -Top row of window  
| BR      integer    -Bottom row of window  
| STATUS   var boolean -Boolean error flag  
| Purpose  
|   Averages the grey level's of the pixels and rewrites them to the  
|   datacube.  
| Logical name assignment  
|   aveproc ::= 'usr.imp.arldir/aveproc'  
| Author: Jim  
|-----  
| Procedure Aveproc (leftcol,rightcol,toprow,bottomrow:integer;  
|                     var status: boolean); external;  
-----
```

```
-----  
| Parameters          AVERAGE -- DRIVER FOR AVEPROC  
| none  
| Purpose  
|   Driver program for the aveproc procedure calls the WINDOW procedure  
|   to determine the area of the data cube to be determined.  
| Logical name assignment  
|   AVERAGE ::= 'usr.imp/average'  
| Author: Jim  
|-----  
| AVERAGE <window>  
-----
```

Appendix A

```
-----  
| Parameters          BRESENHAM -- DRAWS OR ERASES A LINE <LNK .MRL>  
| X1     integer      -x1 coordinate of one endpoint of line  
| Y1     integer      -y1 coordinate of one endpoint of line  
| X2     integer      -x2 coordinate of second endpoint  
| Y2     integer      -y2 coordinate of second endpoint  
| LEVEL   integer    -Grey level of desired line  
| Purpose  
| Given two points, Procedure Bresenham draws (or erases) a line between  
| those two points.      Level > 0 means DRAW line  
|                           Level < 0 means ERASE line  
| Logical name assignment  
| Bresenham ::= 'usr.imp.arldir/bresenham'  
| Author: Bresenham  
|  
| PROCEDURE BRESENHAM (x1,y1,x2,y2,level: integer); external;  
-----
```

```
-----  
| Arguments          CHAINCODE -- DRIVER FOR CHAINPROC <LNK .EXE>  
| Window -- Optional. Default: none. Form: CURSOR or W=X1,Y1,X2,Y2.  
| Use CURSOR to select a window via cursor; use W= to specify actual  
| coordinates. X1,Y1 are coordinates of window top left corner; X2,Y2  
| are coordinates of window bottom right corner.  
| Gray level -- Optional. Default: none. Form: G=n, where n is the  
| gray level that the chain code will follow.  
| Histogram -- Optional. Default: none. Form: HIST or NOHIST.  
| Histogram, if requested, is performed before chain code begins.  
| Purpose  
| Constructs chain code of Datacube image and writes to given file.  
| Logical name assignment  
| CHAINCODE ::= '/usr.imp/chaincode'  
| Author: Dan  
| Example:  
| CHAINCODE W=0,0,100,100 G=20 NOHIST  
|  
| chaincode <window> <gray level> <histogram>  
-----
```

Appendix A

```
| Parameters           CHAINPROC -- SEARCHES FOR PIXEL <LNK .MRL> |
| LC,RC    integer   -Designates left and right column of window.
| TR,BR    integer   -Designates top and bottom row of window.
| Level    integer   -Designates grey level.
| NUMLINKS integer   -Number of links in a chain, usually 30.
| FLAG     var boolean -false if error encountered in procedure.
| DARK     var boolean -indicates whether object is light or dark.
|
| Purpose
|   Searches within the window given for a pixel of the given grey level
|   and follows adjacent pixel links of the appropriate grey level. Links
|   of chain may be sent to printer or designated file.
|
| Logical name assignment
|   CHAINPROC ::= 'usr.iap.arldir/chainproc'
|
| Author: Dan
```

```
| PROCEDURE CHAINPROC (leftcol,rightcol,toprow,bottomrow,level,numlinks:
|                      integer; var flag, dark: boolean); external;
```

```
| Parameters           COMMAND_DB -- SETTS UP DATACUBE <LNK .MRL>
|   COMMAND  char  -Indicates which routine desired for datacube.
|                 Valid possibles are:
|                 'T' (TV)      -Sets datacube to the acquire mode
|                 'F' (FREEZE)  -Freeze's the image on the datacube.
|                 'R' (READ)    -Read black&white from the datacube with
|                               autoincrementation of X coordinate
|                               after each read.
|                 'W' (WRITE)   -Writes black&white to the datacube with
|                               autoincrementation of X coordinate
|                               after each write.
|                 'A' (ACCESS)  -Access the datacube for either reading
|                               or writing without autoincrementation.
|
| Purpose
|   Command_db sets up the datacube for appropriate routines.
|
| Logical name assignment
|   Commanddb ::= 'usr.iap.arldir/commanddb'
|
| Include file: '/usr.iap/iplib.inc'
|
| Author: Barbara
```

```
| PROCEDURE COMMANDDB (command: char; var status: longint); external;
```

Appendix A

```
| Parameters          CURSOR -- DATACUBE CURSOR <LNK .EXE> |
| None
| Purpose
|   Allows user to move a cursor about the Datacube and mark selected
|   points. Permitted commands are:
|     INV      inverts the cursor color
|     DON      to stop the program
|     ST?      to set mark (?=A..Z,0..9,-,+,:, etc.)
|     POS      to write the position
|     MOD      alternates speed between fast and slow
| Logical name assignment
|   cursor ::= '/usr.imp/cursor'
| Author: Barbara
|
|-----|
| CURSOR
```

```
| Parameters          CURSPROC -- CREATES MOVEABLE WHITE CURSOR <LNK .MRL>
|   STARTX  var integer -X coordinate of begining cursor position
|   STARTY  var integer -Y coordinate of begining cursor position
|   MAX     integer -Number of entries desired
|   Xarray,Yarray var array[1..max] of integer
|                   -Coordinates of points returned br procedure
|   Carray   var array[1..max] of char
|                   -Char of pixels at hte points returned
| Purpose
|   Cursproc creates a white cursor at startX and startY of the datacube,
|   and allows usr to move cursor about the datacube with arrows. (See
|   CURSOR for available commands.) The user may designate up to the max-
|   imum number of points to be passed. User exits the program by typing
|   'pos' twice.
| Logical name assignment
|   CURSPROC ::= 'usr.imp.arldir/cursproc'
| Author: Barbara
|
|-----|
| PROCEDURE CURSOR2 (var startX,startY: integer;
|                     var Xarray,Yarray: array[1..<max>] of integer;
|                     var Carray: array[1..<max>] of char); external;
```

Appendix A

```
-----  
| Parameters          CURSPROC2 -- RETURNS DATACUBE COORDS. <LNK .MRL>  
| startX      integer -beginning X coordinate of cursor.  
| startY      integer -beginning Y coordinate of cursor.  
| x1,y1      var integer -Point1 returned to calling program.  
| x2,y2      var integer -Point2 returned to calling program.  
| Purpose  
| Cursproc2 creates a white cursor (at startX/startY of datacube), and  
| allows user to move it about the datacube with arrows. The user may  
| designate two points on the datacube whose x and y coordinates are  
| returned.  
| Logical name assignment  
| Cursproc2 ::= 'usr.imp.arldir/cursproc2'  
| Author: Barbara  
|-----  
| PROCEDURE CURSOR3(startX,startY:integer; var x1,y1,x2,y2:integer); external;  
-----
```

```
-----  
| Parameters          DESTWNDW -- DEFINE A WINDOW <LNK .MRL>  
| Leftcol    var integer -defines leftcolumn of window.  
| toprow     var integer -defines toprow of window.  
| rightcol   var integer -defines right column of window  
| bottomrow  var integer -defines bottom row of window.  
| ARGSTACK    array[0..25] of integer  
|                  -<see procedure getargs for use of argstack>  
| Purpose  
| commands  
| Logical name assignment  
| Destwndw ::= 'usr.imp.arldir/destwndw'  
| Author: JANE  
|-----  
| PROCEDURE DESTWNDW (var x1,y1,x2,y2: integer; Argstack: array[0..25] of  
|                      integer); external;  
-----
```

Appendix A

```
-----  
| Arguments | EDGE -- DRIVER FOR EDGEPROC <LNK.EXE>  
| Formal - None |  
| Interactive - Window |  
| Purpose |  
| EDGE is the driver program for EDGEPROC procedure.  
| Logical name assignment |  
| EDGE ::= 'usr.lsp/edge'  
| Author: Jim |  
|-----  
| EDGE  
-----
```

```
-----  
| Parameters | EDGEPROC -- APPLIES 3X3 CONVOLUTION <LNK.MRL>  
| LC integer | -Left column of desired window |  
| RC integer | -Right column of desired window |  
| TR integer | -Top row of desired window |  
| BR integer | -Bottom row of desired window |  
| MASK array[1..9] of integer | -Convolution mask (3X3) in row-major order |  
| STATUS var boolean | -Status of successful completion of operation |  
| THRESH real | -Percentage of highest valued edge pixels kept |  
| Purpose |  
| Edgeproc applies a user-specified 3X3 convolution mask. It keeps only |  
| the highest THRESH percentage of the resulting edge pixels. The rest are |  
| set to zero. |  
| Logical name assignment |  
| EDGEPROC ::= 'usr.lsp.mrldir/edgeproc'  
| Author: Jim |  
|-----  
| PROCEDURE EDGEPROC (leftcol,rightcol,toprow,bottomrow : integer;  
| mask : array[1..9] of integer; var status : boolean;  
| threshpercentage : real); external;  
-----
```

AD-A153 112 FURTHER STUDY OF DIGITAL MATCHING OF DISSIMILAR IMAGES 2/2
(U) L N K CORP SILVER SPRING MD D LAVINE ET AL FEB 85
ETL-8385 DRCAT76-82-C-0003

UNCLASSIFIED

F/G 12/1

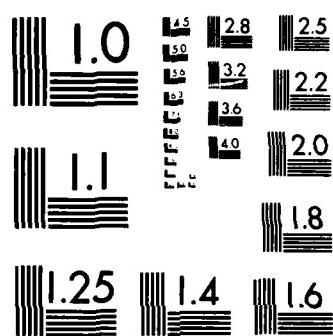
NL



END

FILED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Appendix A

```
-----  
| Arguments           EXPAND -- EXPAND AN IMAGE <LNK .EXE>  
| Source Window -- Default: none. Form: ALL, CURSOR or W=x1,y1,x2,y2.  
|   Use ALL to specify whole image; use CURSOR to select a window via  
|   cursor; use W= to specify actual coordinates of window. X1,Y1 are  
|   coordinates of top left corner of window; X2,Y2 are coordinates of  
|   bottom right corner of window.  
| Destination Window -- Default: upper left corner.  
|   Form: Sfactor, DW=x1,y1,x2,y2 or DC.  
|   Two methods: specify expansion factor or destination window  
|   Use S= to specify expansion factor (integer); then can use DW= to  
|       supply top left corner of window, or DC to use cursor to specify  
|       top left corner of window.  
|   Use DW= to specify both corners of window or DC to use cursor.  
|       dinate of destination window; use DC to use cursor.  
| Background level -- Default: none. Form: B=integer.  
|       Sets outside of destination window to background level. If number > 65:  
|       then no change occurs.  
| Purpose  
|       Expand is the driver program for the EXPANDPROC procedure.  
| Logical name assignment  
|       EXPAND ::= 'usr.imp/expand'  
| Author: Jim  
|-----  
| EXPAND <source window> <destination window> <background level>  
-----
```

Appendix A

```
| Parameters          EXPANDPROC -- EXPAND AN IMAGE <LNK .MRL>
| SLC    integer      -Left column of source window.
| SRC    integer      -Right column of source window.
| STR    integer      -Top row of source window.
| SBR    integer      -Bottom row of source window.
| DLC    integer      -Left column of destination window.
| DRC    integer      -Right column of destination window.
| DTR    integer      -Top row of destination window.
| DBR    integer      -Bottom row of destination window.
| BACKGROUND integer  -Grey level of background.
| FACTOR   integer   -Expansion factor
| PICTURE   var array [0..319, 0..239] of char
|                   -temporary storage
| Purpose
| EXPANDPROC expands a window on the Datacube. If no expansion factor then
| the vertical and horizontal factors are calculated from the source and
| destination window sizes. Pixels are replicated for expansion. If the
| background level is less than 64, the background is filled in.
| Logical name assignment
| EXPANDPROC ::= 'usr.imp.mrldir/expandproc'
| Author: Jim
| PROCEDURE EXPANDPROC (oldleft,oldright,oldtop,oldbottom : integer;
|                       newleft,newright,newtop,newbottom : integer;
|                       background, factor : integer; var picture :
|                           array [0..319,0..239] of char); external;
```

```
| Parameters          FREEZE -- TAKE A PICTURE <LNK .EXE>
| none
| Purpose
| Freezes the image on the datacube.
| Logical name assignment
| FREEZE ::= 'usr.imp/freeze'
| Author: Barbara
| FREEZE
```

Appendix A

```
-----  
| Parameters           GETARGS -- DECODES COMMAND LINE <LNK .INC>  
| ARGSTACK      var array[0..25] of integer; -Each element linked to  
|                   elements in the command line.  
| Purpose  
| ARGSTACK #      ARGUMENT  
| 1,2,3,4        WINDOW=x1,y1,x2,y2  
| 5              CURSOR  
| 6              ALL  
| 7              THRESHOLD=integer  
| 8              LOWERVALUE=integer  
| 9              UPPERVALUE=integer  
| 10             SAMPLINGFACTOR=integer  
| 11,12          DOMAIN=low,high  
| 13,14          RANGE=low,high  
| 15             BACKGROUND=integer  
| 16             HISTOGRAM  
| 17             GREYLEVEL=integer  
| 18,19,20,21    DW=x1,y1,x2,y2 (destination window)  
| 22             DC          (destination cursor)  
| 23             PRINT       (print histograms)  
| 24             VIEW       (output to screen)  
| 25             STRING     (returns string between " ")  
| Logical name assignment  
| Author: Jane and Daniel  
-----  
| Z/usr.imp.source/getargs.inc  
-----
```

```
-----  
| Parameters           GETARGS -- DECODES COMMAND LINE <LNK .INC>  
| ARGSTACK      var array[0..25] of integer; -Each element linked to  
|                   elements in the command line.  
| Purpose  
| Logical name assignment  
| Author: Jane and Daniel  
-----  
| Z/usr.imp.source/getargs.inc  
-----
```

Appendix A

```
| Parameters           HISTCRT -- ANALYZES GREY LEVELS <LNK.MRL> |
| LC      integer    -Left column of desired window.          |
| RC      integer    -Right column of desired window.         |
| TR      integer    -Top row of desired window.            |
| BR      integer    -Bottom row of desired window.          |
| THRESHOLD var integer -Returns optional threshold.        |
| Purpose
|   Hist crt analyzes the grey levels of the pixels on the datacube within
|   the region specified by the window coordinates, and draws a histogram
|   on the terminal screen. for some programs it is desireable to enter a
|   integer value before exiting that will be returned to calling program.
| Logical name assignment
|   HISTCRT ::= 'usr.imp.mridir/histcrt'
| Author: Dan
|
| PROCEDURE HISTCRT (leftcol,rightcol,toprow,bottomrow: integer;
|                     var threshold: integer); external;
```

```
| Arguments           HISTOGRAM -- DRIVER FOR HISTCRT <LNK.EXE> |
| Window -- Optional. Default: none. Form: CURSOR or W=x1,y1,x2,y2.
|   Use CURSOR to select a window via cursor; use W= to specify actual
|   coordinates. X1,Y1 are coordinates of window top left corner; X2,Y2
|   are coordinates of window bottom right corner.
| Print or View -- The user may designate on the command line whether to
|   have the histogram sent to the screen ('V') or the printer ('P').
|   (See getargs).
| Purpose
|   Histogram is the driver program for the HISCRT or the HISTPRT pro-
|   cedures.
| Logical name assignment
|   HISTOGRAM ::= 'usr.imp/histogram'
| Author: Dan
|
| HISTOGRAM <window> <print or view>
```

Appendix A

```
: Parameters          HISTPRT -- ANALYZES GREY LEVELS <LNK .HRL>
: LC      integer   -Left column of desired window.
: RC      integer   -Right column of desired window.
: TR      integer   -Top row of desired window.
: BR      integer   -Bottom row of desired window.
: IDENT    string    -Puts a banner across top of output
:
: Purpose
: Histprt analyzes the grey levels of the pixels on the datacube within
: the region specified by the window coordinates, and sends the histogram
: to the printer. For some programs it is desireable to enter a
: integer value before exiting that will be returned to calling program.
:
: Logical name assignment
: HISTPRT ::= 'usr.imp.arldir/histprt'
:
: Author: Dan
:
:-----
```

```
: PROCEDURE HISTCRT (leftcol,rightcol,toprow,bottomrow: integer;
:                      ident: string); external;
```

```
: Parameters          IMPLIB.INC -- INCLUDES DECLARATIONS <LNK .INC>
: none
:
: Purpose
: Implib.inc includes the declarations of the following external pro-
: cedures:
:           COMMAND_DB (cmd: char; var status: longint); external;
:           SET_XY_DB (var x,y: integer; var status: longint); external;
:           SET_X_DB  (var x: integer; var status: longint); external;
:           SET_Y_DB  (var y: integer; var status: longint); external;
:           READ_DB   (var ch: char; var status: longint); external;
:           WRITE_DB  (var ch: char; var status: longint); external;
:           CURSOR3 (startX,startY: integer; var x1,y1,x2,y2: integer);
:                           external;
:
: Logical name assignment
: Implib ::= 'usr.imp/implib.def/'
:
: Author: Barbara
:
:-----
```

```
: Z/usr.imp.source/implib.inc
```

Appendix A

```
: Parameters          PUTWINDOW -- OUTPUT TO MONITOR <LNK .MRL>
:   LC      integer    -Left column of desired window
:   RC      integer    -Right column of desired window
:   TR      integer    -Top row of desired window
:   BR      integer    -Bottom row of desired window
:   WINDOW  var wind_type -storage for outputted window
:   STATUS   var longint -status of write operation
: Purpose
: Putwindow writes WINDOW onto the monitor at the specified window.
: See WNDOWTYPE.
: Logical name assignment
: PUTWINDOW := 'usr.iap.macro/putwindow.mac'
: Author: BAL
: -----
: PROCEDURE PUTWINDOW (leftcol,rightcol,toprow,bottomrow : integer;
:                      var window : wind_type; var status : longint );
```

```
: Parameters          QNIKHIST -- QUICK HISTOGRAM PROCEDURE <LNK .EXE>
:   LC,RC    integer    -Designates left and right columns of window
:   TR,BR    integer    -Designates top and bottom row of window
:   threshold integer   -Designates threshold returned (if desired)
:   factor    integer    -Designates sampling factor to select pixels
:
: Purpose
: Selects pixels (by increment of sampling factor) within the window and
: provides histogram of this sample of pixels.
: Authors: Dan and Jane
:
: -----
: PROCEDURE QNIKHIST (leftcol,rightcol,toprow,bottomrow: integer;
:                      var threshold,factor: integer); external;
```

Appendix A

```
-----  
: Arguments          QWKHISTGM -- DRIVER FOR QNIKHIST (LNK.MRL) :  
: Window -- Optional. Default: none. Form CURSOR or W=X1,Y1,X2,Y2. :  
: Use CURSOR to select a window via cursor; use W= to specify actual co- :  
: ordinates. X1,Y1 are coordinates of window top left corner; X2,Y2 are :  
: coordinates of bottom right corner. :  
: Samplingfactor -- Optional. Default: none. Form: S=n where n is an inte- :  
: ger. :  
: Logical name assignment :  
: QWKHISTGM := '/usr.iap/qwkhistga' :  
: Author: Dan and Jane :  
: Example: :  
: QWKHISTGM w=50,50,180,200 s=5 :  
:  
:-----  
: qwkhistgm <window> <samplingfactor>  
:-----
```

```
-----  
: Parameters          READ_DB -- RETURNS PIXEL GREY LEVEL (LNK.MRL)  
: CH      var char   -Returns the pixel grey level.  
: STATUS  var longint -Error checking.  
: Purpose  
: Read_db is a pascal callable assembly routine that returns a pixel gray  
: level from the datacube (the position of the pixel is determined prev-  
: iously by the SET_XY_DB procedure).  
: Logical name assignaent  
: Readdb := '/usr.iap.arldir/readdb'  
: Include file: '/usr.iap/iaplib.inc'  
: Author: Barbara  
:  
:-----  
: PROCEDURE READ_DB (var ch: char; var status: longint); external;  
:-----
```

Appendix A

```
-----  
| Arguments          RESCALE -- ALTERATE PIXEL GREY VALUES <LNK .EXE>  
| Window -- Optional. Default: none. Form: Cursor or W=x1,y1,x2,y2.  
| Use Cursor to select a window via cursor; use W= to specify actual  
| coordinates. X1,Y1 are coordinates of window top left corner; X2,Y2  
| are coordinates of window bottom right corner.  
| Purpose  
|   The rescale program allows the user to alterate the pixel grey levels  
|   within a certain DOMAIN to thier corresponding values in a specific  
|   RANGE. If the range low is greater than the range high, the gray levels  
|   will be inverted.  
| Logical name assignment  
|   Rescale ::= '/usr.iap/rescale'  
| Author: Barbara  
|-----  
| RESCALE <window> <domain=<low>,<high>> <range=<low>,<high>>  
-----
```

```
-----  
| Parameters          RESTRWIND -- INPUT FROM STORAGE <LNK .MRL>  
| F      var text    -pointer to file  
| F_NAME string      -file name  
| WINDOW var wind_type -storage for inputted window  
| Purpose  
|   Restrwind reads the specified window from the disk file specified by  
|   F_NAME.  
|   See WINDOWTYPE.  
| Logical name assignment  
|   RESTRWIND ::= 'usr.iap.macro/restrwind.sac'  
| Author: BAL  
|-----  
| PROCEDURE RESTRWIND (var f : text; f_name : string;  
|                      var window : wind_type);  
-----
```

Appendix A

```
| Parameters          RSCLPROC -- STRETCH/SHRINK GREY LEVELS <LNK .MRL> |
| LC     integer    -defines left column.                                |
| RC     integer    -defines right column.                               |
| TR     integer    -defines top row.                                    |
| BR     integer    -defines bottom row.                                 |
| DL     integer    -The low domain.                                    |
| DH     integer    -The high domain.                                 |
| RL     integer    -The low range.                                   |
| RH     integer    -The high range.                                 |
| Purpose
| Procedure RSCLPROC 'stretches' or 'shrinks' the grey levels within the
| the domain over the range.
| Logical name assignment
|   Rsclproc :=='/usr.iop.arldir/rsclproc'
| Author: Barbara
|-----
| Procedure Rsclproc (leftcol,rightcol,toprow,bottomrow,domainlow,
|                      domainhigh,rangelow,rangehigh: integer); external;
```

```
| Arguments          SAMPLE -- DRIVER FOR SAMPLPROC <LNK .EXE>
| Window -- Optional. Default: none. Form: Cursor or W=X1,Y1,X2,Y2.
|           Use Cursor to select a window via cursor; use W= to specify actual
|           coordinates. X1,Y1 are coordinates of window top left corner; X2,Y2
|           are coordinates of window bottom right corner.
| Purpose
| Sample is the driver program for samplproc. It allows the user to
| reduce a specific are of the datacube either by a sampling factor, or
| by designating the dimensions of the destination window.
| Logical name assignment
|   Sample :== '/usr.iop/sample'
| Author: Daniel
|-----
| SAMPLE <window> <samplingfactor=<integer> or dw=x1,y1,x2,y2 or dc>
|           <background=<integer>>
```

Appendix A

```
| Parameters          SAMPLPROC -- REDUCE AN IMAGE <LNK .MRL>
| oldright,oldleft,oldttop,oldbottom   integer -defines old window.
| newleft,newright,newtop,newbottom   integer -defines new window.
| factor                   integer -background factor.
| picture                  var array[0..319,0..239] of char
|                         -Contains image on the datacube.
|
| Purpose
| Samplproc reduces the image in the old window and rewrites it to the
| new window. If desired, the background may be filled in.
|
| Logical name assignment
| Samplproc ::= '/usr.iap.mrldir/samplproc'
|
| Author: Daniel
|
| PROCEDURE SAMPLPROC (oldleft, oldright, oldtop, oldbottom, newleft,
|                      newright, newtop, newbottom, background, factor:
|                      integer; var picture: array[0..319,0..239] of char);
|                      external;
```

```
| Arguments          SCALE -- WORD PROC. AID <LNK .EXE>
| Mode              - ON or OFF. This creates or erases a number line.
| Col. Pointer      - Must be two digits, sets flashing arrow at that point.
|
| Purpose
| Scale is a word processing aid, which creates a number line at the
| bottom of the screen, or erases a previously existing line. If ON, an
| optional integer in the range 1 to 80 may be given to specify a special
| column pointer.
|
| Examples:
| SCALE ON       -Prints a scale from 1 to 80 along bottom of screen.
| SCALE OFF      -Removes a currently displayed scale from screen.
| SCALE ON 40    -Prints a scale from 1 to 80 along bottom of screen, where
|                   the 40th column has a flashing arrow instead of a digit.
|
| Logical name assignment
| SCALE ::= 'usr.misc/scale'
|
| Author: Michael
|
| SCALE <mode> <column pointer>
```

Appendix A

```
| Parameters           SET_X_DB -- DESIGNATE X COORD ON CUBE <LNK .MRL> |
| X      var integer   -Designates the x coordinate on the datacube. |
| STATUS var longint   -Error handling. |
| Purpose              |
| Set_x_db is a pascal callable assembly routine that designates the x |
| coordinate of a pixel on the datacube. |
| Logical name assignment |
| SETXDB := 'usr.iap.arldir/setxdb' |
| Include file: '/usr.iap/implib.inc' |
| Author: Barbara |
|-----|
| PROCEDURE SET_X_DB (var x: integer; var status: longint); external; |
```

```
| Parameters           SET_XY_DB -- DESIGATE PIXEL ON CUBE <LNK .MRL> |
| X      var integer   -designate x coordinate on datacube. |
| Y      var integer   -designate y coordinate on datacube. |
| STATUS var longint   -error handling. |
| Purpose              |
| Set_xy_db is a pascal callable assembly routine that designates the x |
| and y coordinates of a pixel on the datacube. |
| Logical name assignment |
| SETXYDB := 'usr.iap.arldir/setxydb' |
| Include file: '/usr.iap/implib.inc' |
| Author: Barbara |
|-----|
| PROCEDURE SET_XY_DB (var x,y: integer; var status: longint); external; |
```

```
| Parameters           SET_Y_DB -- DESIGNATE Y COORD. ON CUBE <LNK .MRL> |
| Y      var integer   -Y coordinate on datacube. |
| STATUS var longint   -Error handling. |
| Purpose              |
| Set_y_db designates the y coordinate of a pixel on the datacube. |
| Logical name assignment |
| SETYDB := 'usr.iap.arldir/setydb' |
| Include file: '/usr.iap/implib.inc' |
| Author: Barbara |
|-----|
| PROCEDURE SET_Y_DB (var y: integer; var status: longint); external; |
```

Appendix A

```
-----  
| Parameters           STOREWIND -- OUTPUT TO STORAGE <LNK .MRL> |  
| F      var text    -pointer to file  
| WINDOW  var wind_type -storage for outputted window  
| Purpose  
|   Storewind writes the specified window to the disk file pointed to by F.  
|   F must already have been set prior to the procedure call.  
|   See WINDOWTYPE.  
| Logical name assignment  
|   STOREWIND ::= 'usr.imp.macro/storewind.sac'  
| Author: BAL  
|-----  
| PROCEDURE STOREWIND (var f : text; var window : wind_type);  
-----
```

```
-----  
| Arguments           THRESHOLD -- DRIVER FOR THRPROC <LNK .EXE> |  
| Window -- Optional. Default: none. Fore: Cursor or W=x1,y1,x2,y2.  
|   Use CURSOR to select a window via cursor; use W= to specify actual  
|   coordinates. X1,Y1 are coordinates of window top left corner; X2,Y2  
|   are coordinates of window bottom right corner.  
| Purpose  
|   Used to convert any Datacube image into a binary image.  
| Logical name assignment  
|   THRESHOLD ::= 'usr.imp/threshold'  
| Author: Dan  
|-----  
| THRESHOLD <window> <threshold=<integer>> <lower value=<integer>>  
|   <upper value=<integer>>  
-----
```

Appendix A

```
| Parameters           THRPROC -- THRESHOLD AN IMAGE <LNK .MRL>
| LC      integer     -Left column of desired window
| RC      integer     -Right column of desired window
| TR      integer     -Top row of desired window
| BR      integer     -Bottom row of desired window
| LEVEL   integer     -Threshold level
| LOW     integer     -New grey level below threshold
| HIGH    integer     -High grey level above threshold
|
| Purpose
| Thrproc thresholds the specified window. Pixels with grey levels below
| the threshold LEVEL will be set to the new grey level LOW (if LOW is > 63)
| they will be unchanged). Pixels with grey levels above or equal to LEVEL
| will be set to HIGH. If HIGH is > 63 then pixels with grey levels above
| the threshold will be unchanged and pixels at the threshold will be set
| to LOW.
|
| Logical name assignment
| THRPROC ::= 'usr.imp.mrldir/thrproc'
|
| Author: Dan
|
|-----|
| PROCEDURE THRPROC (leftcol,rightcol,toprow,bottomrow : integer;
|                     level, lowvalue, highvalue : integer ) ; external;
```

```
| Arguments           TV -- OPENS DATA_CUBE FOR IMAGES <LNK .EXE>
| none
|
| Purpose
| TV allows the datacube to accept images from the camera.
|
| Logical name assignment
| TV ::= 'usr.imp/tv'
|
| Author: Barbara
|
|-----|
| TV
```

Appendix A

```
-----  
| Parameters           WINDOW -- SPECIFY WINDOW ON MONITOR <LNK .MRL>  
| LC      var integer    -Left column of desired window  
| RC      var integer    -Right column of desired window  
| TR      var integer    -Top row of desired window  
| BR      var integer    -Bottom row of desired window  
| Purpose  
| Window allows the user to specify a window on the monitor using either  
| the cursor or by supplying the coordinates or by indicating the whole  
| image. The choice must have been indicated on the command line of the  
| calling program. Essentially this routine just checks GETARGS for you.  
| See GETARGS.  
| Logical name assignment  
|   WINDOW ::= 'usr.imp.arldir/window'  
| Author: Jane  
|-----  
| PROCEDURE WINDOW (var leftcol,rightcol,toprow,bottomrow : integer);  
|   external;  
-----
```

```
-----  
| Type Declaration     WINDOW TYPES -- USED TO WORK IN MEMORY <LNK .INC>  
| ROW_TYPE      array [0..239] of integer;  
| WIND_TYPE     record  
|   num_columns  : 0..319;  
|   num_rows     : 0..239;  
|   grey_level   : array [0..319] of row_type  
| end;  
| Purpose  
| These global type declarations are necessary in order to use the routines  
| which require the storage of windows in memory.  
| Logical name assignment  
|   ZZ/usr.imp.macro/type.inc  
| Author: BAL  
|-----  
| ZZ/usr.imp.macro/type.inc  
-----
```

```
-----  
| Parameters          WRITE_DB -- WRITES PIXEL GREY LEVEL <LNK .MRL>  
| CH      var char   -Returns pixel grey level to be written.  
| Status   var longint -Error checking.  
| Purpose  
|   Write_db is a pascal callable assembly routine that writes a pixel of  
|   the grey level specified by CH to the datacube (the x and y coordinates  
|   previously having been designated).  
| Logical name assignment  
|   Writedb := 'usr.iap.arldir/writedb'  
| Include file: '/usr.iap/iaplib.inc'  
| Author: Barbara  
|-----  
| PROCEDURE WRITE_DB (var ch: char; var status: longint); external;  
-----
```

```
-----  
| Parameters          WRITEPIX -- WRITE A PIXEL TO DATACUBE <LNK .MRL>  
| X      integer     -X coordinate of point to be written.  
| Y      integer     -Y coordinate of point to be written.  
| CH  var char       -Grey level of point to be written.  
| Purpose  
|   Writepix is a pascal callable assembly routine that writes a pixel to  
|   the datacube at the point specified by x and y.  
| Logical name assignment  
|   Writepix := 'usr.iap.arldir/writepix'  
| Author: Barbara  
|-----  
| PROCEDURE WRITEPIX (x,y: integer; ch: char); external;  
-----
```

END

FILMED

5-85

DTIC